АЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра Автоматики і управління у технічних системах

«На правах рукопису»
УДК _____

«Допущений до захисту»
Завідувач кафедри
_____ А.І.Ролік
(підпис) (ініціали, прізвище)
«____» _____ 2018 р.

Магістерська дисертація

зі спеціальності (спеціалізації) 126 Інформаційні системи та технології
(код і назва спеціальності)

на тему: Активаційні функції радіально-базисних нейронних мереж на ПЛІС

Виконав студент II курсу, групи ІА-73мп
(шифр групи)

Юрченко Захар Олегович _____
(прізвище, ім'я, по-батькові) (підпис)

Керівник старший викадач кафедри АУТС Шимкович В.М. _____
(посада, наукова ступінь, звання, прізвище, ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, наукова ступінь, звання, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, наукова ступінь, вчене звання, прізвище, ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 р.

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

ФІОТ

Кафедра АУТС

Ступінь вищої освіти «магістр»

Зі спеціальності 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Ролік А.І.
(підпис) (ініціали, прізвище)
«__» _____.

ЗАВДАННЯ
на магістерську дисертацію студенту

Юрченку Захару Олеговичу .
(прізвище, ім'я, по батькові)

1. Тема роботи: Активаційні функції радіально-базисних нейронних мереж на ПЛІС ,

керівник роботи ст. викладач кафедри АУТС Шимкович Володимир Миколайович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____ р. №

2. Строк подання студентом проекту _____

3. Вихідні дані до магістерської дисертації чип «Spartan-3», абсолютна похибка не більше 0.005, функція Гауса

4. Перелік завдань, які потрібно розробити: огляд, аналіз та опис предметної галузі і готових рішень, метод апаратно-програмної реалізації штучних нейронних мереж, алгоритм апаратної реалізації штучного нейрона з сигмоїдальною функцією активації, розробка алгоритму апаратної реалізації нейронів прихованого шару RBF-мережі, проведення дослідів з різними чипами ПЛІС, проведення дослідів з різними архітектурами нейромереж, аналіз стартап-проекту

5. Перелік графічного матеріалу блок-схема радіально-базисної нейронної мережі, блок-схема алгоритм, функція Гауса, апаратна реалізація RBF-мережі на ПЛІС,

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 29.10.2018

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Строк виконання етапів проекту	Примітка
	Огляд, аналіз та опис предметної галузі і готових рішень	29.10.2018	
	Розробка алгоритму апаратної реалізації штучного нейрона з сигмоїдальною функцією активації	07.11.2018	
	Розробка алгоритму апаратної реалізації нейронів прихованого шару RBF-мережі	15.11.2018	
	Проведення дослідів з різними чипами ПЛІС, проведення дослідів з різними архітектурами нейромереж	21.11.2018	
	Проведення аналізу стартап-проекту	26.11.2018	
	Оформлення ПЗ	04.12.2018	

Студент _____ Юрченко З.О.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Шимкович В.М.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Магістерська дисертація освітньо-кваліфікаційного рівня “магістр” на тему: «Активаційні функції радіально-базисних нейронних мереж на ПЛІС». Робота включає сторінок, рисунків, таблиць, додатків та джерел.

У магістерській дисертації було проведено апаратну реалізацію радіально базисних-нейронних мереж на ПЛІС та розроблений її алгоритм. За рахунок паралельних обчислень, це дозволяє значно підвищити швидкодію роботи мереж, при мінімальному використанні обчислювального ресурсу.

За темою даної роботи був проведена доповідь на конференції ICACIT-2017 у Кракові (додаток А).

Радіально-базисна нейронна мережа, функції активації

ABSTRACT

ЗМІСТ

ВСТУП

У наш час стрімко розвивається наука та техніка. Разом з покращенням обчислювальних пристроїв ускладнюються і задачі, які на них покладають. Теорію автоматичного управління це не обійшло стороною. Раніше методи автоматичного управління могли спиратися на теорію лінійних систем, але зараз цього не достатньо, бо реальні об’єкти нелінійні за своєю природою. Нейронні мережі вважають нелінійними динамічними системами. Це сучасний напрямок в теорії автоматичного управління. При подаванні вхідної інформації паралельно разом з можливістю навчання нейронних мереж дасть високу швидкодію та зробить цю технологію бажаною для побудов систем з автоматичним управлінням [1].

Радіально-базисні нейронні мережі швидко навчаються та мають здатність моделювати довільну нелінійну функцію. Тому їх часто використовують для вирішення задачах класифікації, у прогнозуванні часових рядів, у системах управління та функціях наближення.

У наші дні для реалізації нейронних мереже у системах управління використовують програмний метод, використовуючи комп'ютерну техніку або спеціалізовані контролери, які побудовані на їх основі. Вартість таких регуляторів досить значна для використання їх в простих системах управління, що дуже звужує область застосування і робить їх недоцільними. До цього ж комп'ютерні регулятори на нейронних мережах вимагають значний час навчання та їх продуктивність обмежена. Коли нейронні мережі працюють послідовно та рекурентно по всій множині параметрів, вони не встигають за темпом динамічного об’єкта, тому проблема швидкодії лишається невирішеною для навчання навчання нейронних мереж. І тому для вирішення даної проблеми вводять паралельну роботу внутрішніх елементів нейромереж та процедуру навчання. Програмовані логічні інтегральні схеми(ПЛІС) мають можливість розпаралелювати процеси, і тому є актуальною реалізація на них нейронних мереж.

Метою роботи є підвищення швидкодії роботи та покращення точності обчислень радіально-базисних нейронних мереж, розпаралелишви обчислення, при їх апаратній реалізації на ПЛІС.

Задачі:

- Дослідити існуючі рішення в області реалізацій нейронних мереж
- Реалізувати декілька моделей радіально-базисних нейронних мереж на ПЛІС
- Провести дослідження впливу будови радіально-базисних нейронних мереж на швидкодію
- Провести дослідження впливу будови радіально-базисних нейронних мереж обчислювальний ресурс ПЛІС.

Об’єкт досліджень виступають радіально-базисні нейронні мережі

Предмет досліджень є методи та алгоритми апаратної реалізації функцій активації радіально-базисних нейронних мереж

ПЕРЕЛІК СКОРОЧЕНЬ

БНМ – багатошарова нейронна мережа

ЕОМ – електронно-обчислювальна машина

НДР – науково дослідна робота

ОЗП – оперативний запам’ятовуючий пристрій

ПКМ – програмований комбінаційний пристрій

ПЛІС – програмована логічна інтегральна схема

ПМ – програмований мультиплексом

ТР – тригер затримки

ШНМ – штучна нейронна мережа

FPGA(англ. Field-Programmable Gate Array) – програмована користувачем вентильна матриця, ПКВМ

LUTs (Look Up Table) – вентилі логічної матриці

RBF-мережа – нейронна мережа з радіально базисними функціями

VHDL (англ. VHSIC (Very high speed integrated circuits) Hardware Description Language) – мова програмування, якою розробляють апаратуру сучасних обчислювальних систем.

1 ОГЛЯД ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Програмована логічна інтегральна схема – як елементна база нейронних обчислювачів

Програмована логічна інтегральна схема (далі ПЛІС) – являє собою електронний компонент, який використовується для створення інтегральних цифрових схем. На відміну від логічного елемента, який має фіксовану функцію, ПЛІС має невизначену функцію під час виготовлення. Перед тим як ПЛІС може бути використана в схемі вона повинна бути запрограмована [2].

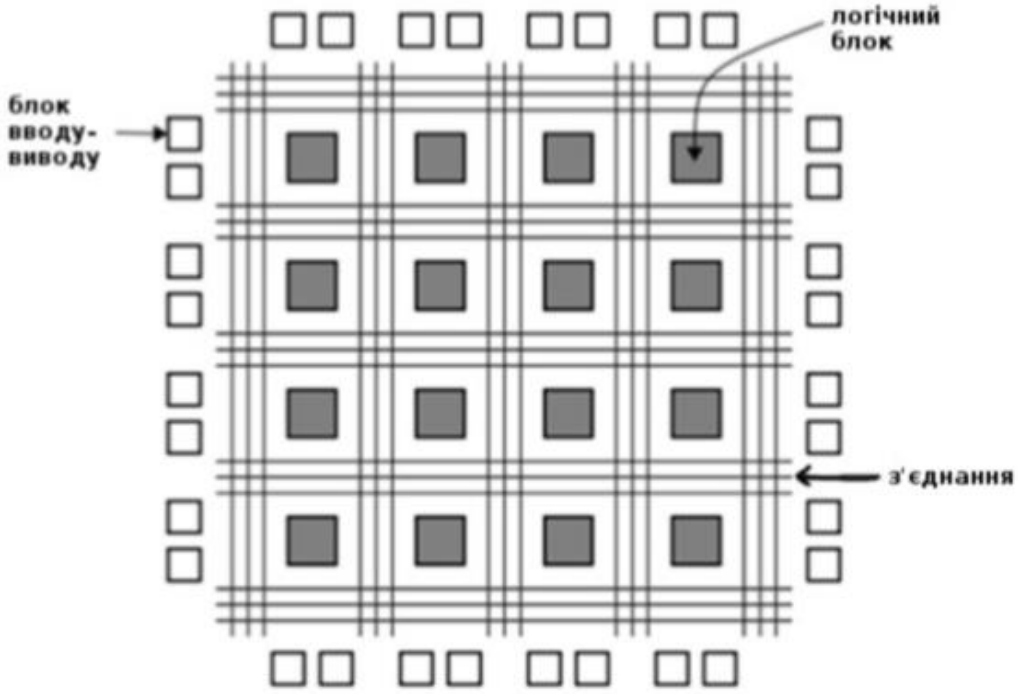


Рисунок 1.1 – Архітектура ПЛІС

На сьогоднішній день ПЛІС випускаються промисловістю усього світу, тому вони мають досить широку номенклатуру. ПЛІС пройшли за короткий час (близько п’ятнадцяти років) ПЛІС пройшли у своєму розвитку шлях від простих пристроїв, еквівалентних кільком цифрових інтегральних схем середнього ступеня інтеграції,

до надвеликих інтегральних схем з еквівалентним об'ємом порядку понад десяти мільйонів логічних елементів.

Основні типи ПЛІС мають різні модифікації схемної побудови:

- програмована користувачем вентильна матриця(FPGA);
- комплекс логічного програмованого пристрою(CPLD);
- програмовані матриці логіки(PAL);
- загальні логіки масиву(GAL).

Проте ці типи ПЛІС мають декілька загальних принципів побудови та функціонування дивись рисунок 2.1.

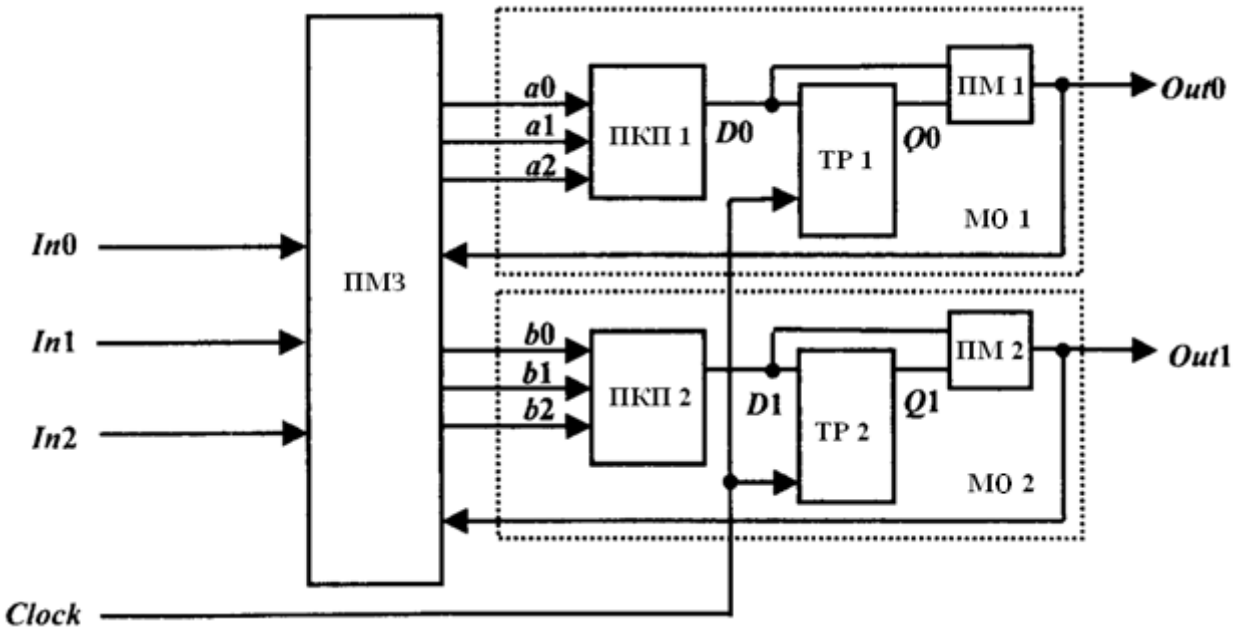


Рисунок 2.1 – Структурна схема ПЛІС

Із рисунку 2.1 видно, що ПЛІС складається з вузлів макроосередків МО1 і МО2, що обведені пунктиром і програмованої матриці з'єднань (ПМЗ). На пристрій подається чотири входи, три з яких (In0, In1 і In2) логічні, а четвертий – Clock, який подає тактовий сигнал. Виходи пристрою (Out0 і Out1) – це виходи макроосередків. На даному прикладі присутні лише два макроосередки, але у сучасних ПЛІС їх кількість досягає кількох десятків тисяч.

Макроосередки (МО) – це основні операційні блоки ПЛІС, кожен з яких має:

- програмований комбінаційний пристрій (ПКП);
- тригер затримки (ТР)
- програмованого мультиплексора (ПМ).

Програмований комбінаційний пристрій (ПКП) призначається для реалізації комбінаційних цифрових пристроїв. В даному прикладі вони мають три входи (a2, a1, a0/b2, b1, b0) та один вихід (DO/D1). Здійснюється функціонування таких пристроїв логічними функціями.

Кожна функція програмується розробником. Коли завантажуються конфігурації ПЛІС, визначені розробником функції вводяться в ПКП, після чого вони стають комбінаційними схемами, що виконують запрограмовані логічні функції.

Тригер затримки (ТР1, ТР2) запам'ятовує значення сигналів ПКП, коли на нього подається тактовий вхід сигналу Clock. Тригер в макроосередку дозволяє побудову цифрових автоматів з пам'яттю(лічильників, регістрів та ін.).

Програмований мультиплексор (ПМ) передає сигнал на вихід макроосередку вихідного сигналу тригера або безпосередньо сигналу з ПКП. У першому випадку макроосередок виконує функції цифрового пристрою з пам’яттю, а у другому – комбінаційного цифрового пристрою.

ПЛІС широко використовується для побудови різних за складністю і за можливостями цифрових пристроїв [3]. Наприклад:

- пристроїв з великою кількістю портів введення-виведення;
- пристрої для виконання цифрової обробки сигналу;
- пристрої для виконання передачі даних на високих швидкостях;
- пристрої для виконання криптографічних операцій, системи захисту інформації;
- пристрої, які допомагають проектувати інтегральні схеми спеціального призначення;
- пристроїв для виконання ролі мостів (комутаторів), що поєднують системи, які мають різну логіку або різну напругу живлення;

- реалізація нейрочипів;
- пристрої для виконання моделювання квантового обчислення.

Серед альтернатив ПЛІС існують такі приклади:

- БМК (Uncommited Logic Array) – базові матричні кристали, які є напівфабрикатом інтегральних схем, що виробляються в масових кількостях без орієнтації на конкретного споживача, містять типовий набір елементів - транзисторів, резисторів, та на заключних етапах виготовлення на заводі програмуються технологічно, шляхом нанесення маски з'єднань останнього шару металізації;
- ASIC (application-specific integrated circuit) – інтегральна мікросхема, що робиться на замовлення з програмним забезпеченням для вирішення конкретного завдання;
- спеціалізовані процесори або мікроконтролери (по вільно іше) мікроелектронні програмовані пристрої, призначені для використання в керуючих вузлах різноманітних технічних виробів, системах передачі даних і системах управління технологічними процесами, проте працюють більш повільно за ПЛІС.

Архітектура ПЛІС побудована із трьох таких головних елементів:

а) програмовані логічні блоки, складовими яких є настроювані логічні блоки, об'єднані в масив (для забезпечення функціональних елементів) та виконує більшу частину логіки ПЛІС. Будь-який логічний блок обладнаний двома перемикачами, реалізовує будь-яку комбіновану логічну функцію з п’ятьма входами;

б) програмовані з'єднувальні ресурси, що встановлюють маршрут єднання серед логічних блоків, що потребують налаштування;

в) блоки входу-виходу, що необхідні для забезпечення інтерфейсу серед монтажних контактів ПЛІС та внутрішніх сигнальних магістралей. Можливо програмування як вхід чи вихід, або двонаправлений порт.

Наразі при реалізації нейрообчислювачів використовують гібридні схеми - блок матричних обчислень реалізують на базі кластерного поєднання процесорів

цифрових сигналів, а логіку управління - на основі ПЛІС. Надалі матричне ядро реалізують на базі нейрочипів, а сигнальні процесори і ПЛІС – залишають в якості основи для побудови логіки управління. Наразі багато фірм в світі розроблює та випускає різні ПЛІС, але, першість займають дві фірми – Xilinx і ALTERA. Пріоритетність фірм визначити неможливо тому, що технічні характеристики продукції майже не відрізняються [4].

На рисунку 1.2 представлено плати Cyclone 2 фірми ALTERA, яка місить приблизно 5 тисяч обчислювальних таблиць.

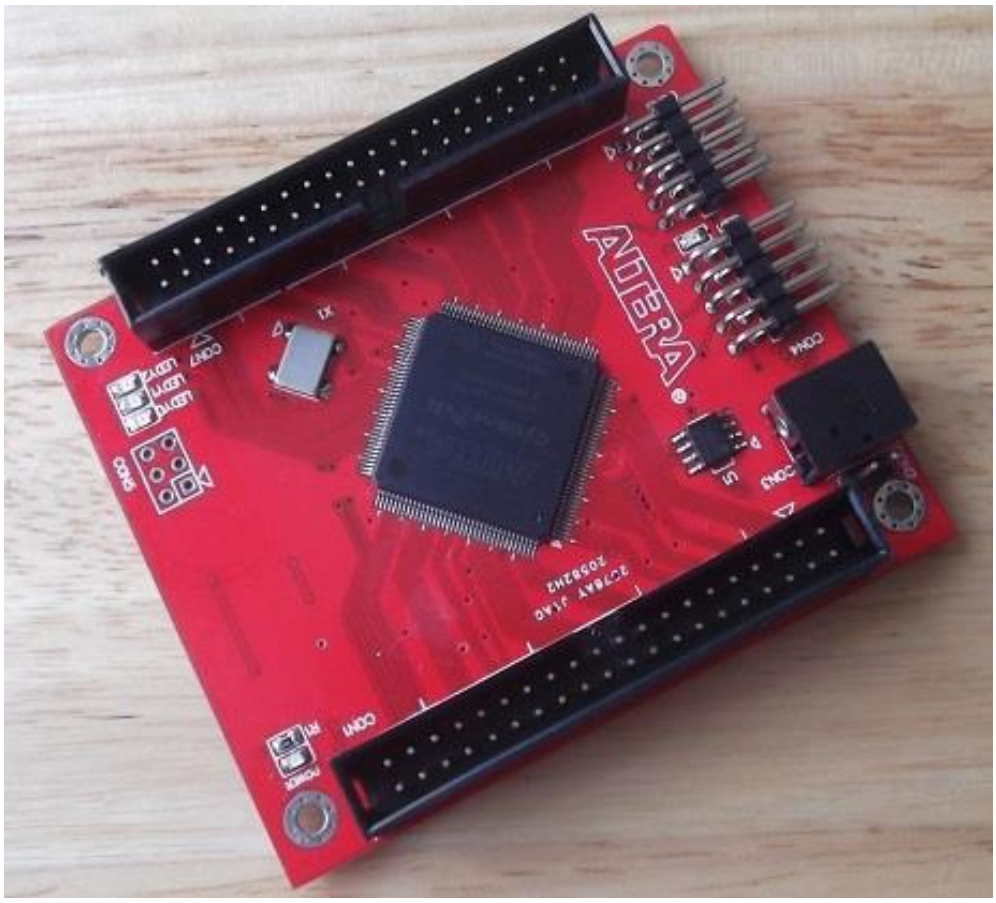


Рисунок 1.2 – Altera CycloneII EP2C5T144 [5]

Фірма Xilinx випускає схожі за зовнішнім виглядом ПЛІС схожу на таку, яку випускає фірма ALTERA. ПЛІС фірми Xilinx має такі характеристики:

- висока продуктивність з системними частотами досягає 300 МГц;
- значний обсяг ресурсів може мати до 4 млн. системних вентилів на кристал;

- наявна можливість верифікації та ініціалізації через JTAG;
- технологічні норми – до 0.18 мкм на шести шарах металу;
- технологічні норми складають до 0.18 мкм на шести шарах металу;
- номенклатура кристалів широка за типом виконання;
- існує можливість програмувати безпосередньо у системі;
- доступна вартість;
- низьке споживання енергії;
- недорогі та розвинені засоби проектування;
- швидкий час компіляції та малий цикл проектування;
- можливість перекладати проекти в схеми Xilinx на замовлення.

Фірма Xilinx при виготовленні ПЛІС використовуються такі три основні технології:

- основа SRAM типу FPGA, при якому конфігурація ПЛІС зберігається у внутрішньому захованому ОЗУ, а для ініціалізації можна потрапити через зовнішній масив пам'яті. Даною технологією користувались при розробці чипів серії: Spartan, Virtex, XC3000, XC4000, XC5200.
- основа FLASH типу CPLD. В даному випадку конфігурація зберігається у внутрішній незалежній FLASH-пам'яті і тому у будь-який момент часу може бути перевантажена безпосередньо з PC. Даною технологією користувались при розробці чипів серії XC9500.
- основа EEPROM типу CPLD, у цьому випадку конфігурація також зберігається у внутрішній незалежній EEPROM-пам'яті та у будь-який момент часу можна перевантажити безпосередньо з ПЕОМ. Даною технологією користувались при розробці чипів серії CoolRunner.

Програмне забезпечення при проектуванні ПЛІС має три головні компоненти: синтезатор логіки, фіттер та асемблер.

Компілятор аналізує користувацький проект, різні схеми та текстові описи на мовах Verilog HDL або VHDL, та генерує netlist – список всіх елементів схеми та їх зв’язки. Нетліст мусить бути оптимізованим – логічні функції як правило мінімізують, а також видаляють дубльовані регістри.

Фіттер (fitter) мусить вміщувати всю логіку з netlist у наявну архітектуру ПЛІС. Його задача розмістити логічні елементи ти виконати трасування зв’язків між ними. Складність задачі полягає у тому, що один проект може розміщуватись в ПЛІС різними способами, а їх безліч. Деяке трасування та розміщення виявляється кращими, а інші гіршими. Головним критерієм якості отриманої системи виступає максимальна частота, з якою зможе працювати проект при утвореному розміщенні елементів та при даному трасуванні зв’язків. На це впливає кількість програмованих комутаторів між логічними блоками та довжина зв'язків між ними.

1.2 Штучний нейрон та штучні нейронні мережі

Штучні нейрони – це спрощені моделі біологічних нейронів, їх прототипи. Під ними мають на увазі вузли штучної нейронної мережі (ШНМ). Нейрон приймає певну множину сигналів і генерує з них вихідний сигнал, який потім подає або на вихід, або на вхід наступних нейронів. У другому варіанті коли нейрони між собою з’єднані, вони утворюють ШНМ. На рисунку 1.3 представлено модель штучного нейрона.

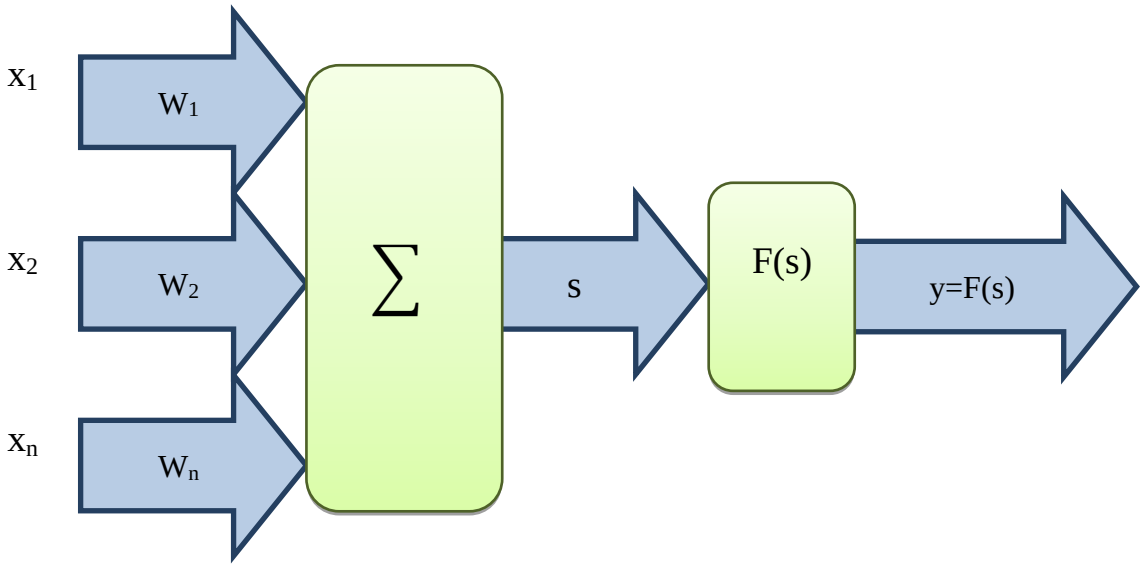


Рисунок 1.3 – Модель штучного нейрона

В цій загальній моделі усі входи множаться на відповідний ваговий коефіцієнт. Потім усі сигнали потрапляють до суматора, де визначається рівень активації нейрона. Біль детально дія нейрона описана у третьому розділі. Спершу буде пояснено про передавальні функції нейронів або ще функції активації та їх варіативність.

Функція активації *f(x)* відображає залежність вихідного сигналу нейрона від суми зважених вхідних сигналів. Для нелінійної роботи нейрона та використовують різні передавальні функції.[6]

Порогова передавальна функція являється перепадом, яка зображена на рисунку 1.4. Якщо вхідне значення більше порогового, то значення функції активації дорівнює максимально допустимому, у іншому випадко – мінімально допустимому.

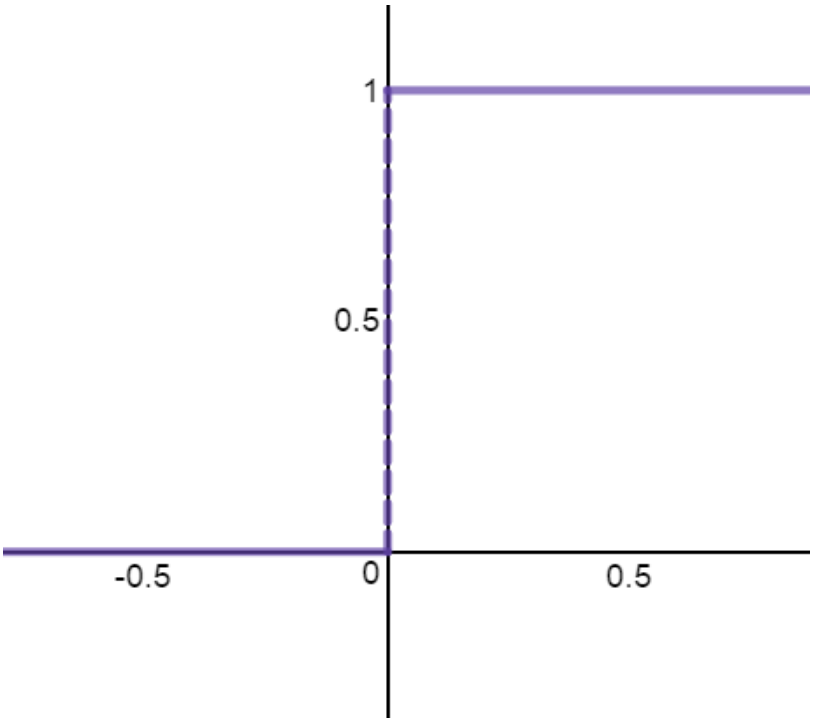


Рисунок 1.4 – Порогова функція активації

Лінійна передавальна функція представлена на рисунку 1.5. Тут присутні дві лінійні ділянки, в яких функція активації тотожно дорівнює максимально

допустимому та мінімально допустимому значенню і є ділянка, на якому функція строго монотонно зростає.

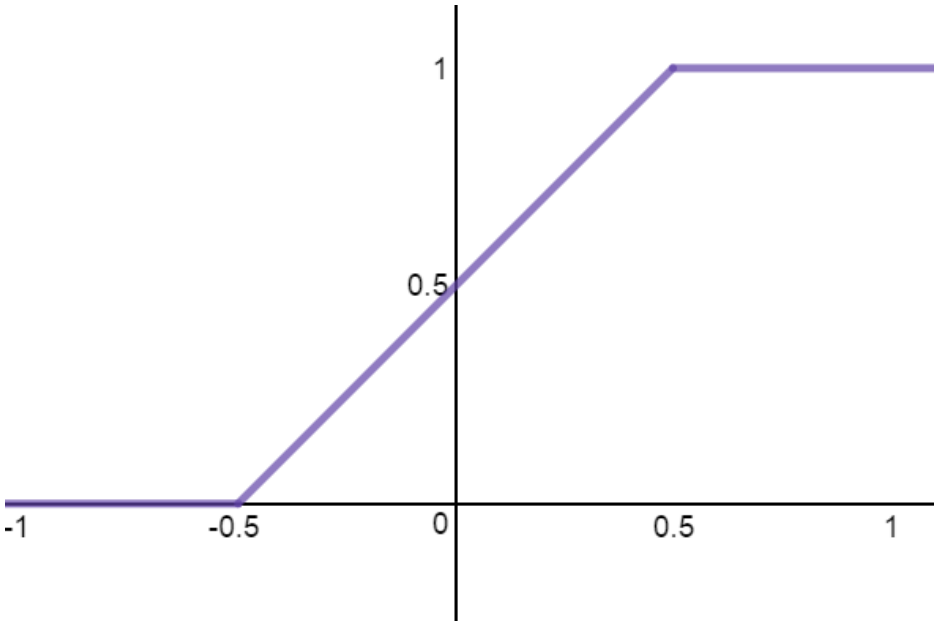


Рисунок 1.5 – Лінійна функція активації з начисленням

Обмеженість нейронних мереж із пороговою функцією активації нейронів зумовив введення функцій сигмоїдального типу. Як представлено на рисунку 1.6, активація відбувається плавно, це дає змогу перейти від бінарних виходів до аналогових. Особливість сигмоїдальної функції у тому, що вона підсилює слабкі сигнали та не насичується від сильних.

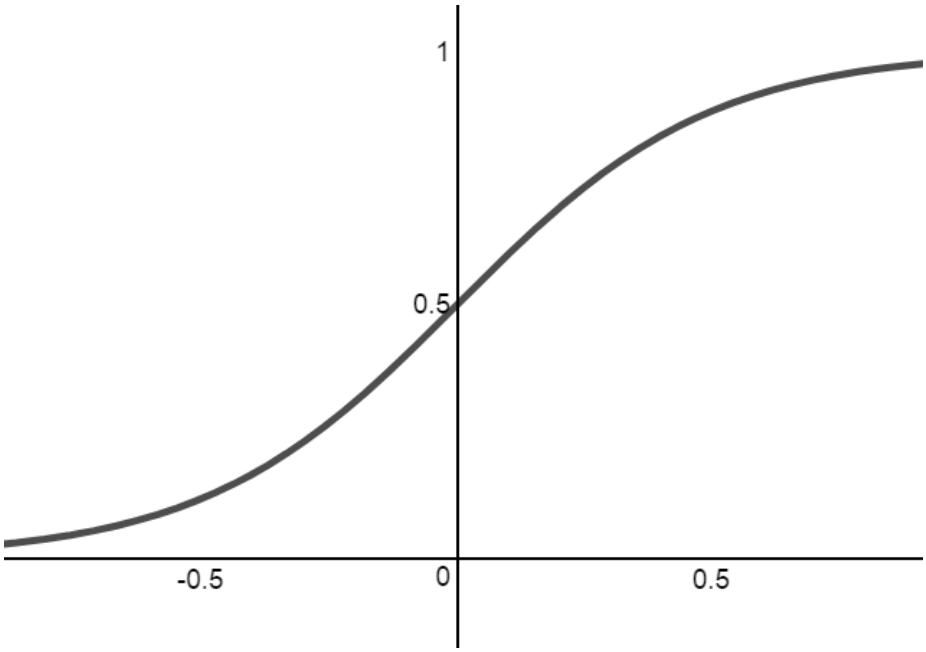


Рисунок 1.6 – Сигмоїдальна функція активації

Задати сигмоїдальний сигнал можна різними способами, прикладами можуть бути логістична функція та гіперболічний тангенс, які відрізняються лише масштабом осей.

Функції активації типу одиничного стрибка та лінійного порогу зустрічаються рідко. Наразі сигмоїдальну функцію активації використовують найчастіше.

Отже, зв’язані між собою нейрони утворюють нейронну мережу. Ті в свою чергу відрізняються своєю архітектурою: структурою зв’язків між нейронами, кількістю шарів, функцією активації нейронів, алгоритмом навчання [1].

Біологічні нейрони утворюють нейронні мережі використовуючи різноманітні з’єднання. Проте для реалізації штучних нейронних мереж існують фізичні обмеження.

Нейрони об’єднуються у мережі та утворюють системи обробки інформації. Такі моделі адаптуються до постійних змін зовнішнього середовища. В процесі їх функціонування відбувається процес перетворення вхідного вектора сигналів у вихідний. А який буде вид перетворення залежить від архітектури нейронної

мережі, засобами керування та синхронізації інформаційних потоків між нейронами і характеристиками самих нейронів.

Важливим фактором ефективності ШНМ є установлення кількості нейронів та типів зв'язків між ними.

При описі нейронних мереж часто використовують такі терміни:

- структура – спосіб зв'язків нейронів у ШНМ;
- архітектура – структура ШНМ та типи нейронів;
- парадигма – спосіб навчання та використання.

Може бути реалізовано нейромережі різної структури з однаковою парадигмою і навпаки.

Групу нейронних мереж на рисунку 1.7, в якій кожен нейрон у мережі зв’язаний лише з сусідніми нейронами, називають слабозв’язними (персептрон, нейронні мережі прямого поширення, автокодувальники та ін.).

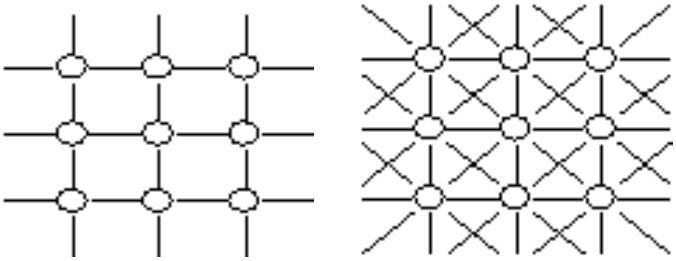


Рисунок 1.7 – Слабозв'язані нейромережі

А мережі, в якій вихід кожного нейрона зв’язаний зі входами усіх інших, як зображено на рисунку 1.8 називають повнозв’язні нейронні (мережі Хопфілда, мережі Кохокена).

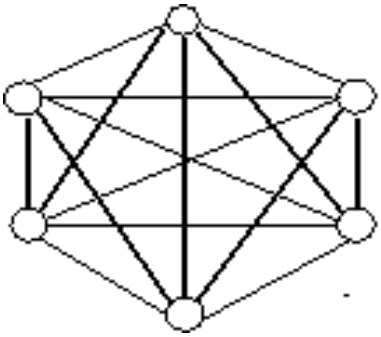


Рисунок 1.8 – Повнозв'язана нейромережа

Найпоширенішим із варіантів архітектури є багатошарові мережі, що зображений на рисунку 1.9. В даному випадку нейрони об'єднуються у шари, які мають єдиний вектор сигналів входів. Зовнішній вхідний вектор подається на рецептори, вхідний шар нейронної мережі. Виходами нейронної мережі є ефектори, вихідні сигнали останнього шару. Крім вхідного та вихідного шарів, ШНМ має один або кілька прихованих шарів нейронів. Нейрони прихованого шару не мають контактів із зовнішнім середовищем.

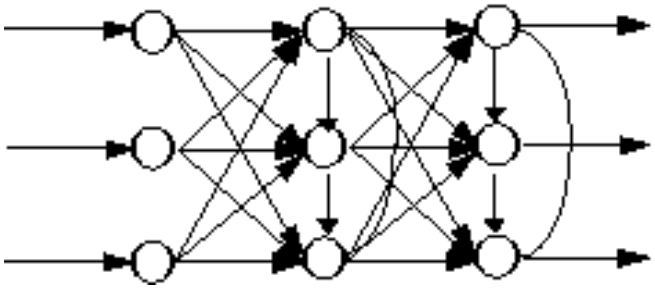


Рисунок 1.9 – Багатошарова нейронна мережа

Напрямок зв'язку від одного нейрону до іншого є важливим аспектом нейронних мереж. В більшості нейромереж кожен нейрон прихованого шару отримує сигнали від всіх нейронів попереднього шар. Після виконання операцій над вхідними сигналами, нейрон передає свій вихід до всіх нейронів наступного шару, забезпечуючи передачу вперед на вихід, такі нейронні мережі називають мережами прямого поширення.

Бувають ще мережі зі зворотнім зв'язком, в яких сигнал на виходів з нейронів скеровується до нейронів попереднього шару (рисунок 1.10).

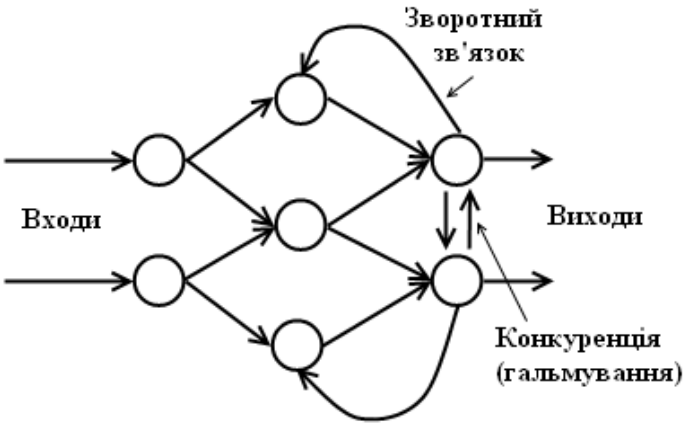


Рисунок 1.10 – Рекурентна мережа

Шлях, яким з’єднуються нейрони між собою має значний вплив на роботу мережі.

1.3 Математика нейронної мережі радіально базисних функцій

Радіально-базисна нейронна мережа – це штучна нейронна мережа на основі радіально базисних функцій. Їх використовують у широкому колі задач, таких як апроксимація, класифікація даних, у прогнозуванні часових рядів, у функціях наближення.

Основна властивість радіально базисної функції – це монотонні зміни і їх відгуки, симетричні щодо деякої вертикальної осі симетрії. Найчастіше використовують функцію Гауса (рисунок 1.11).

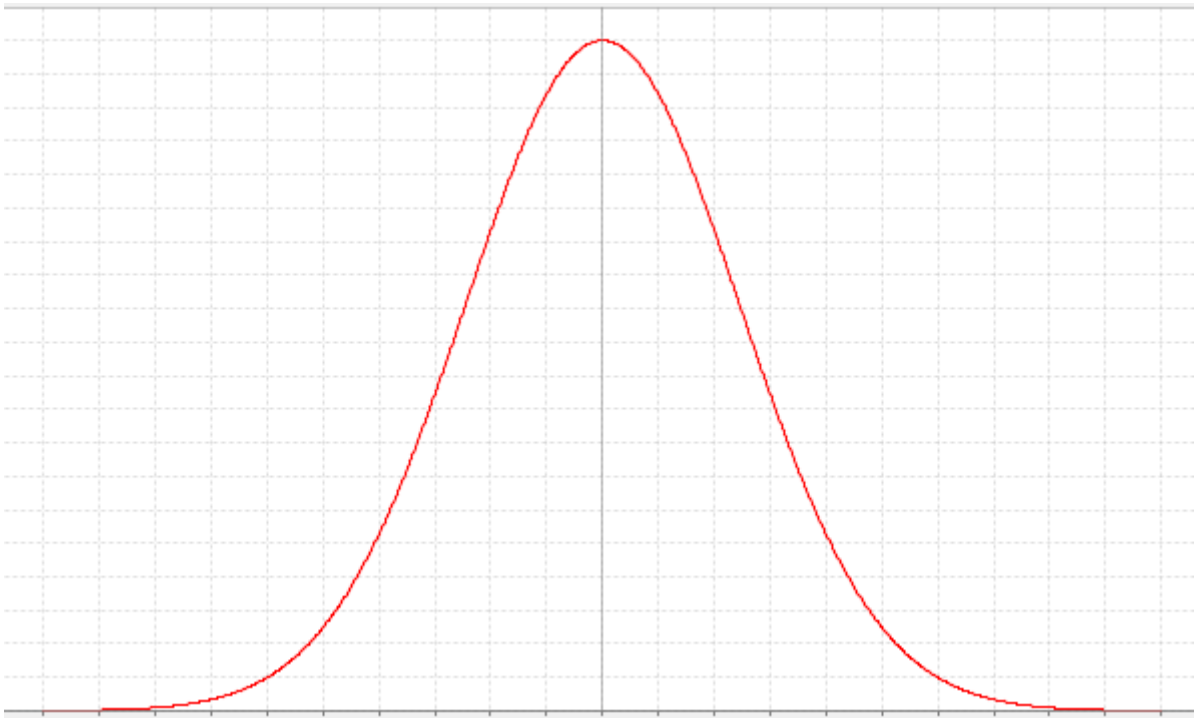


Рисунок 1.11 – Функція Гауса

На рисунку 1.12 наведена структура штучної нейронної мережі на основі радіально-базисних функцій. Функція Гауса описується наступною формулою:

$$y = ae^{-\frac{(x-b)^2}{2\sigma^2}} \tag{1.1}$$

де a – коефіцієнт висоти;

b – зміщення від початку координат;

σ – відповідає за швидкість спадання.

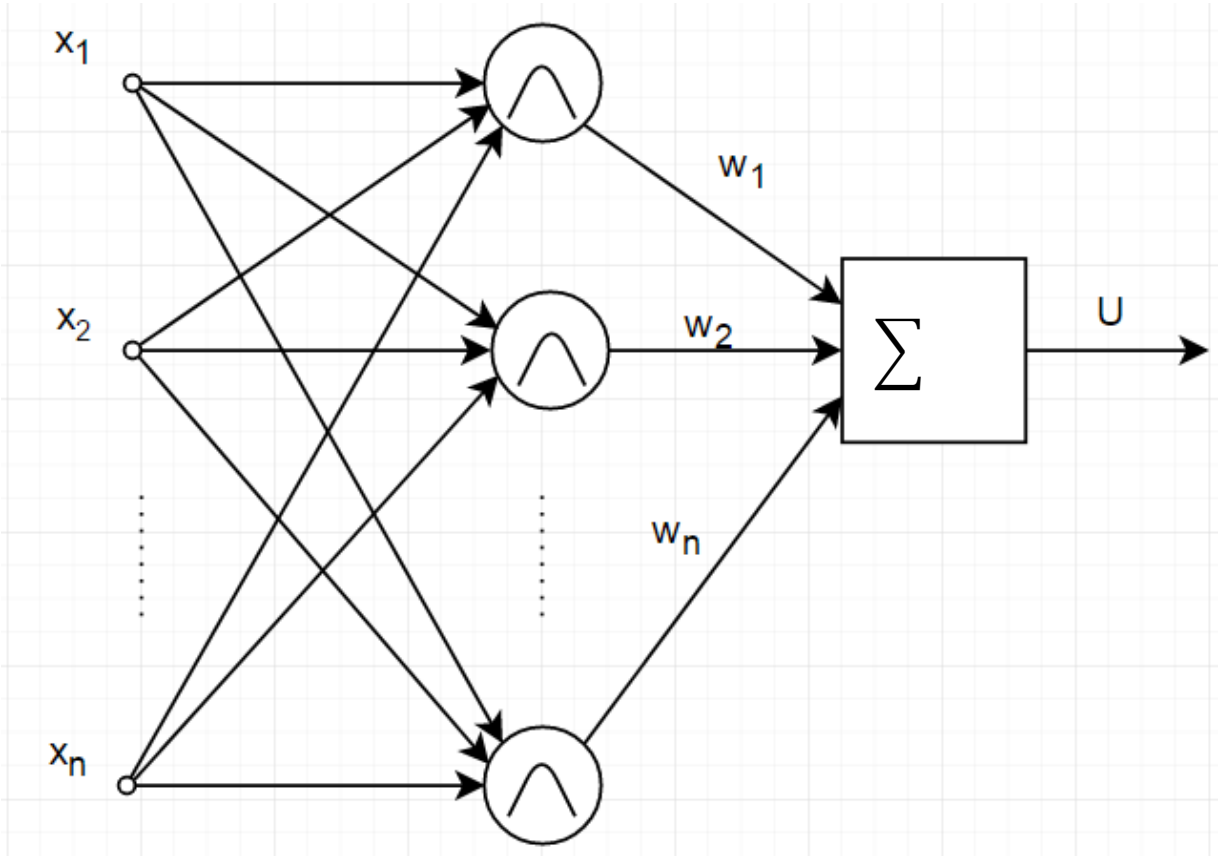


Рисунок 1.12 – Радіально-базисна нейронна мережа

Дана структура містить два шари нейронів. Перший прихований шар активують функції Гауса. Фактично вони обробляють вектор вхідних значень, визначаючи ступінь близькості кожного з них до центрів радіально-базисних функцій. Виходи нейронів другого шару – це лінійні комбінації виходів першого шару [7; 8].

Склад і кількість входів і виходів визначаються класом розв'язуваної задачі. При апроксимації даних входи – це аргументи апроксимуючої залежності, а виходи повертають її значення. При класифікації даних входи – це характерні ознаки, за якими розрізняються об'єкти, що відносяться до кластерів або класам, а виходи вказують на відповідний входам кластер або клас.

Кількість прихованих елементів також залежить від розв'язуваної задачі. Якщо це апроксимація даних, воно може бути будь-яким. У разі кластеризації та класифікації даних кількість нейронів повинні відповідати кількості кластерів [8].

Життєвий цикл штучних нейронних мереж на основі радіально-базисних функцій, як і для більшості інших архітектур, включає дві стадії: навчання і практичного використання.

У свою чергу, на стадії навчання можна виділити також два етапи:

- налаштування нейронної мережі
- оптимізація синаптичних коефіцієнтів лінійного вихідного шару.

На етапі налаштування нейронної мережі визначається центри і радіуси елементів прихованого шару.

При наявності невеликої кількості навчальних прикладів центри можна обрати по відповідним векторам. А при наявності великої кількості прикладів, центром може бути центр потенційного кластера [9].

Вибір радіусів радіальних елементів визначається необхідним видом радіально-базисної функції. Це впливає на швидкість спадання функції. При одних значеннях радіусу графік функції занадто гострий, а це значить, що мережа не буде коректно інтерполювати дані між відомими точками на досить великій відстані від них, так як втрачає здатність до узагальнення навчальних даних. А при інших значеннях радіусу, графік стає пологим, що робить мережу несприйнятливою до окремих деталей.

Тому радіус задають одним з двох способів: математично обчислюють вручну, обраховується автоматично по прикладам навчання.

На етапі оптимізації вагових коефіцієнтів обчислюються:

а) характеристична матриця значень навчальних прикладів

б) матриця вагових коефіцієнтів вихідного шару нейронів.

Серед переваг даної архітектури нейронних мереж виділяють:

- наявність єдиного прихованого шару, достатнього для моделювання яскраво виражених нелінійних залежностей;
- простота алгоритму оптимізації вагових коефіцієнтів;

- гарантоване знаходження глобального оптимуму функції помилки при знаходженні вагових коефіцієнтів нейронів вихідного шару;
- висока швидкість навчання.

До обмежень або недоліки нейронні мереж на основі радіально-симетричні функції можна віднести:

- необхідність спеціального налаштування параметрів радіально-базисної функції;
- неможливість екстраполяції моделі за межами вихідного інтервалу зміни вхідних значень навчальної вибірки.

2 ДОСЛІДЖЕННЯ ІСНУЮЧИХ РІШЕНЬ

У минулому розділі було розглянуто в загальному, якою може бути нейронна мережа та те, що вони можуть займати різну кількість обчислювального ресурсі і мати різний час навчання. Головне, що нейромережі бувають різноманітних типів, мати різну архітектуру, структуру та спосіб навчання. Це все впливатиме на швидкодію мереж та ресурс обчислювального ресурсу і деякі приклади будуть розглянути далі у розділу.

2.1 Процес відбору партій інтегральних схем за допомогою нейронних мереж

Одна з задач, яку можуть виконувати радіально-базисні нейронні мережі – це задача класифікації. Так і в даному випадку нейромережі використовують для пошуку степені надійності інтегральних схем. Партії були поділені на три групи за стелінню надійності: нижчі за технічні вимоги, ті, які відповідають технічним вимогам, вищі за технічні вимоги. У даному джерелу відбувається відбір інтегральних схем, які вище технічних вимог [10].

Відбір проходив за трьома параметрами, за вихідною напругою низького рівня, вихідного струму високого рівня та вихідного струму низького рівня. Всі три величини подаються на один нейрон, який має один вихід, як представлено на рисунку 2.1. Це задача, яку звели до розпізнавання образів. Після 45-и зроблених вимірів, найгірші подавались у систему

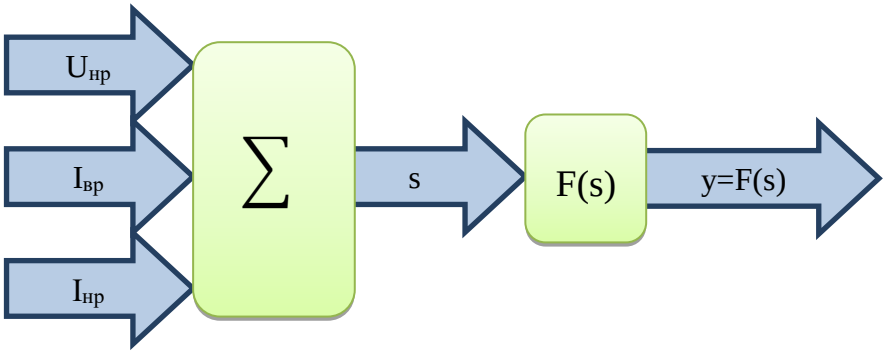


Рисунок 2.1 – Модель нейрона для задачі класифікації

Для задачі по дослідженню більш масштабних інтегральних схем, які мають багато вузлів, було використано широкі нейромережі, що мали двадцять нейронів на вході. На нейрони вхідного шару подаються дані від різних вузлів. Також там же способом досліджували процес деградації вузлів.

У даному прикладі не було даних про швидкодію мереж, бо це була не задача керування у реальному часі. Зате видно один з прикладів використання нейронних мереж та залежність їх архітектури в залежності від поставлених задач.

2.2 Використання нейронних мереж для оцінки якості телекомунікаційних послуг

Щоб оцінити якість інтегральних схем використовувались штучні нейронні мережі всього двох шарів. А в наступному прикладі оцінюється якість телекомунікаційних послуг, для якої використана нечітка логіка в нейромережевому логічному базисі, яка складається з п’яти шарів (рисунок 2.2).

Нейрони вхідного шару, як в інших ШНМ, отримують дані на вхід та інтерпретують їх, тобто виконують функцію рецепторів.

А останній п’ятий шар є вихідним, нейрони на ньому виконують регулюючу функцію.

Більш цікаві нейрони четвертого і другого рівнів, бо вони мають різні функції активації, для надання значень, щоб отримати відповідні лінгвістичні змінні.

Нейрони на третьому шарі користуються нечітким логічним правилом, їх задача на вхід отримати причини, а на вихід видати наслідки.

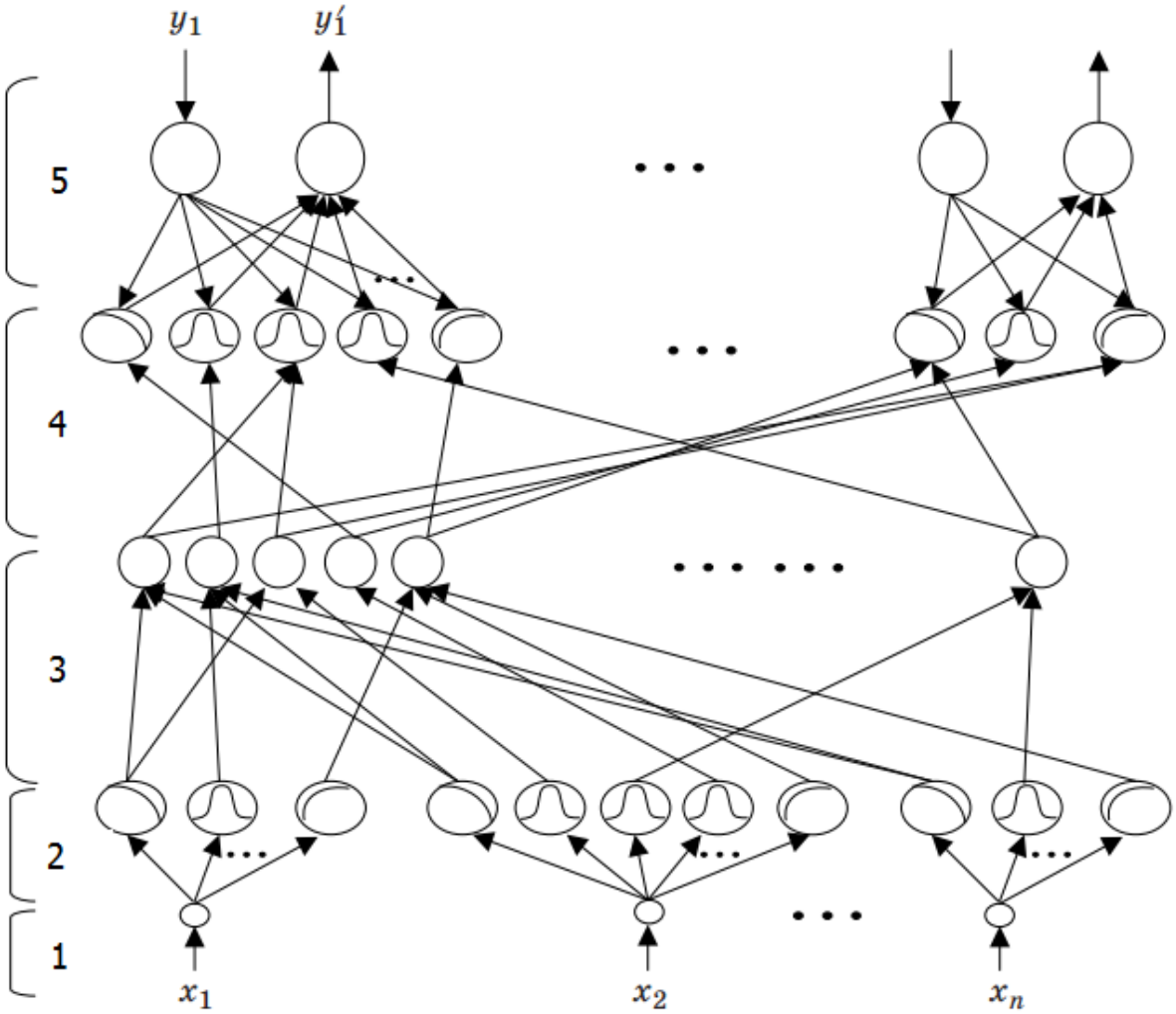


Рисунок 2.2 – П’ятишарова ШНМ[11]

Причинно-наслідкові зв’язки утворюються четвертим, третім і другим рівнями ШНМ. Дана нейромережа має парадигму прямого поширення, бо вона без зворотних зв’язків. На вхід подавались п’ять параметрів: джиттер (випадкові спотворення сигналу), затримка в передачі IP-пакета, імовірність помилки при передачі та імовірність втрати IP-пакета.

Далі вихідні результати було поділено розробниками на певну кількість рівнів, які ділились за якістю передачі

Також у даному прикладі також була утворена таблиця входів і виходів для навчання мережі, але це уже не було розглянуто.

2.3 Нейронні мережі, які реалізовані на багатоядерних процесорах

Один зі способів реалізації ШНМ це реалізація їх на процесорах. Для прикладу розглядається використання багатоядерного процесора SEAforth з різними способами подання нейронів [12].

У даному прикладі нейромережа уже навчена раніше і тому відомі ваги зв’язків. Зовнішня пам’ять містить в собі відомі параметри, а під час початку роботи, коли виконується код відбувається завантаження їх у ядра. Ваги зв’язків можуть зберігатись на стеках або в оперативній пам’яті.

На рисунку 2.3 представлено загальний алгоритм обчислення значення на виході нейрона (додаток Г).

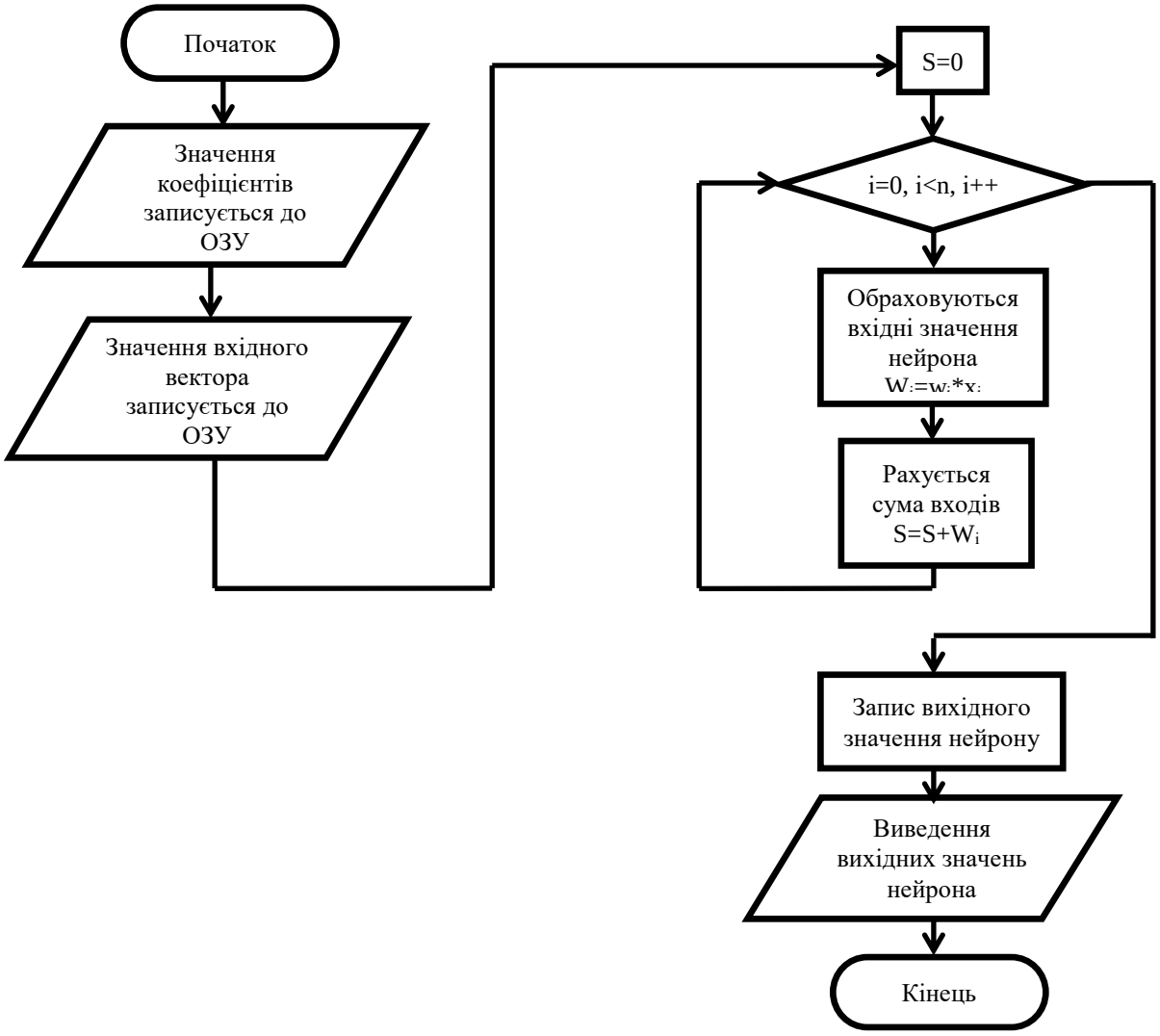


Рисунок 2.3 – Алгоритм обчислення вихідного значення нейрона

Було розглянуто декілька варіантів реалізації нейронних мереж.

В першому прикладі кожен вузол ШНМ представляється повноцінним нейроном з «і» входами. Коли коефіцієнти зв’язків розміщуються в стеку, може бути реалізовано 7-8 зв'язків. Ваги зв’язків також можна розміщувати в оперативній пам’яті, але тоді їх кількість також залежатиме від складності алгоритму обчислення, а саме від функції активації, а також від розрядності коефіцієнтів. Тому в загальному випадку кількість зв’язків, на які можна розраховувати, буде 10-20 штук.

Вхідний вектор також може оброблятись не одним способом. Він може послідовно надходити через певний порт або може бути повністю записаним в оперативну пам’ять.

Ресурс процесора обмежений його технічними характеристиками, тому при повному завантаженні розрахунками процесор фірми SEAFORTH міг працювати з нейромережою не більше ніж на 40 нейронів. Виходить, що є можливість реалізації багатошарової мережі. В такому способі процесор у певний проміжок часу представляє заданий шар нейромережі. Для багатошарової ШНМ будуються унікальні шляхи для передачі даних, бо відповідним способом модифікуються слова, які їх реалізують.

В другому прикладі розглядався випадок, коли шар штучної нейронної мережі обчислюється ядром. Такий варіант добре використовувати для реалізації досить добре використовувати для реалізації мережі, з великою кількістю шарів та нейронів у кожному шарі. Це дозволяє зв’язкам між шарами бути унікальними, що робить специфічним потік даних. При цьому розробник може докласти зусиль, щоб мати змогу для вирішення іншої задачі використовувати вільні ядра та ядра, які вже пропрацювали один шар. Але при такій реалізації використовується багато пам’яті для збереження даних вхідного вектора та коефіцієнтів, а також збільшує час затримки для обчислення виходу шару.

В третьому прикладі більшість ядер процесора використовується для обчислення нейрона, а інші ядра – як конвеєр для руху потоку даних: вхідні та вихідні дані і вагові коефіцієнти. Число синоптичних зв’язків залежить від кількості

синапсів нейрона, в такій схемі їх може бути близько 15. Виходить частина ядер відповідає за ці синоптичні, інші ядра накопичують та підсумовують результат, а також є ще ядра або ядро, чия задача обчислити передавальну функцію нейрона. Така схема має перевагу відносно інших прикладів, бо процесорні ядра постійно завантажені. Це означає, що обчислювальні ресурси раціонально використовуються на повну можливість.

У даному прикладу реалізована схема, коли кожне ядро є унікальним нейроном. В такій схемі персептрон, який має чотири входи обчислювався приблизно 800нс. У порівнянні з більшістю нейрочипів, це досить хороший показник.

Далі буде розглянуто синтез ШНМ.

Ефективним використанням обчислювальних ресурсів можна вважати метод, коли жодне ядро процесора не простоює, а беруть участь у обчисленні штучної нейронної мережі.

Процес обчислення ШНМ, де використовується процесор SEAforth S40C18, представлений на рисунку 2.4.

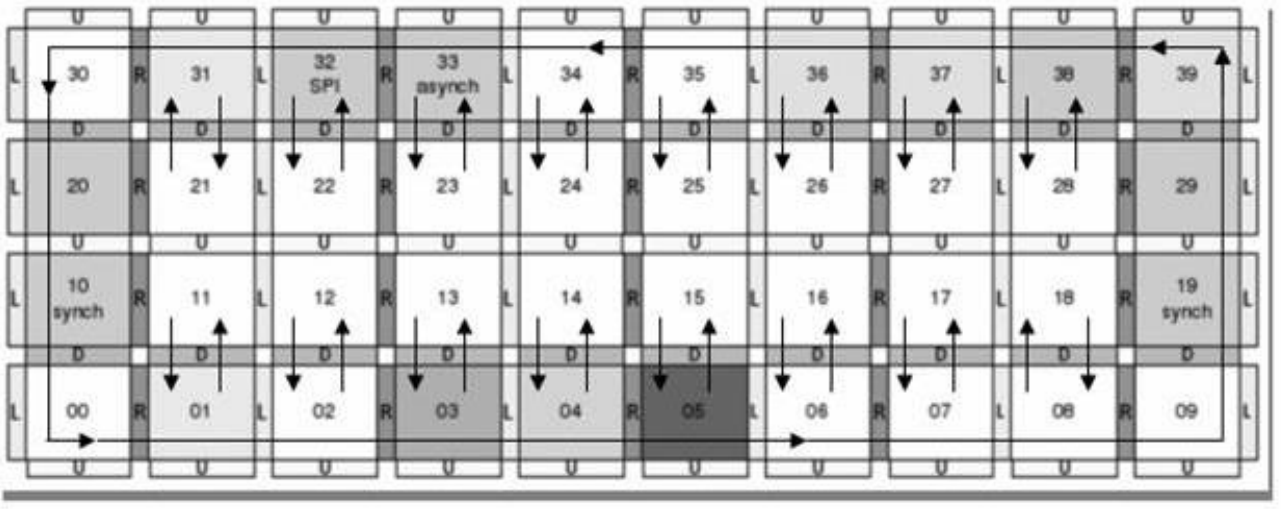


Рисунок 2.4 – Процес обчислення ШНМ на процесорі [12]

Ядра під номерами 11-18 та 21-28 являються нейронами кожного шару нейромережі. А інші ядра забезпечують передачу вихідних сигналів попереднього

шару наступному, тобто утворюють такий собі конвеєр. На рисунку цей процес зображено стрілками, коли потік даних від попереднього шару передається наступному. Зазначалось, що перевага даного методу полягає у можливості програміста реалізувати будь-яку архітектури ШНМ.

Нейронна мережа обчислюється наступним загальним алгоритмом:

а) вхідний вектор $\bar{x}$ розмірністю n – поступає у ядро під номером 10 через послідовний синхронний порт, а тоді в ядро №00;

б) вектор $\bar{x}$ далі з ядра №00 по-бітно надходить до «конвеєру», за яким дані надходять на нейрони і-го шару мережі, ядра 11-18 та 21-28, в залежності до архітектури реалізованої мережі;

в) після цього отриманий вектор $\bar{y}$ як вихідний надходить до «конвеєру», після чого формується у ядрі №00;

г) згідно архітектури мережі, використовуючи інтерфейс SPI можна завантажити в ядра код програми (наприклад, інша передавальну функцію, іншу структуру руху потоку даних і т.п.) або навіть необхідні вагові коефіцієнти синапсів наступного шару мережі;

д) далі вектор, отриманий на попередньому кроці, надходить з ядра №00 на «конвеєр» і тоді до нейронів наступного шару;

е) і вкінці, після того як отримано вихідний вектор мережі, він надходить на ядро №10 а потім до зовнішнього світу за допомогою послідовного синхронного порту.

Виходить, змінюючи код певного стану автомата, програміст отримує змогу реалізувати найрізноманітніші зв’язки між вузлами ШНМ.

2.4 Використання нейронних мереж в бортових системах космічної техніки

Розглянемо приклад реалізації багатошарових персептронів зі чотириступінчастою пороговою функцією активації, що зображено на рис. 2.5, для розв’язання задач цифрової обробки інформації при роботі бортових систем космічних апаратів [13].

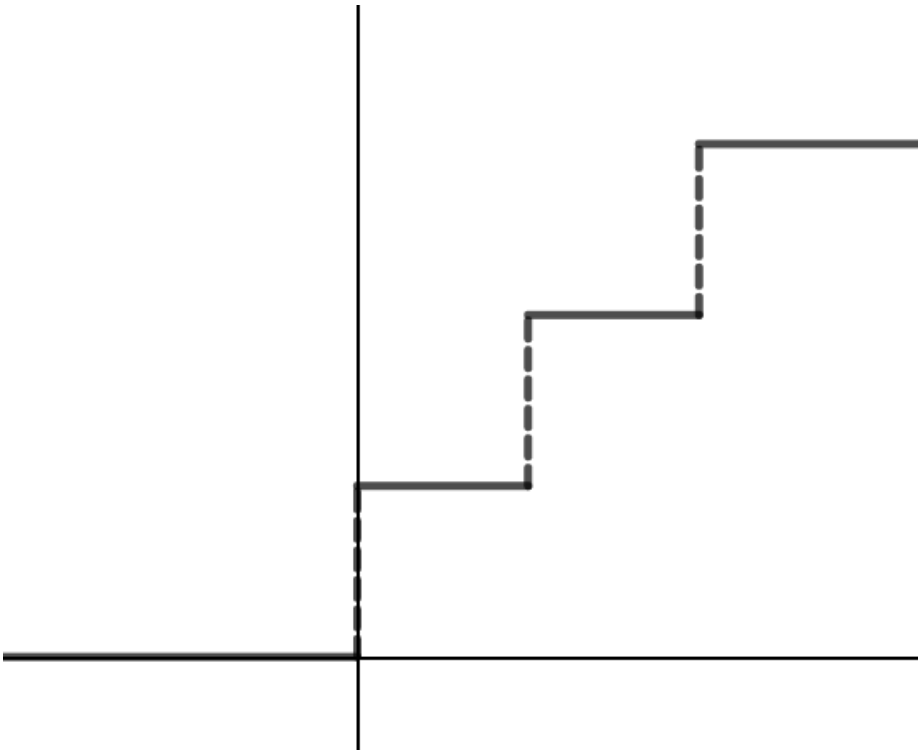


Рисунок 2.5 – D-тригерна порогова функція

Розроблено 3 моделі багатошарових персептронів з 18 вхідними нейронами та різною кількістю проміжних шарів. Перша модель містить 8 проміжних шарів, друга – 16, а третя – 23. Кожен проміжний шар складається з 18 нейронів.

Якщо порівняти її з розглянутими раніше прикладами, то треба зауважити, що дана нейронна мережа зростає за рахунок кількості внутрішніх шарів нейронів при тому, що кількість нейронів одного шару залишається незмінною. На рисунку 2.6 зображена багатошарова нейронна мережа з вісімнадцятьма вхідними значеннями.

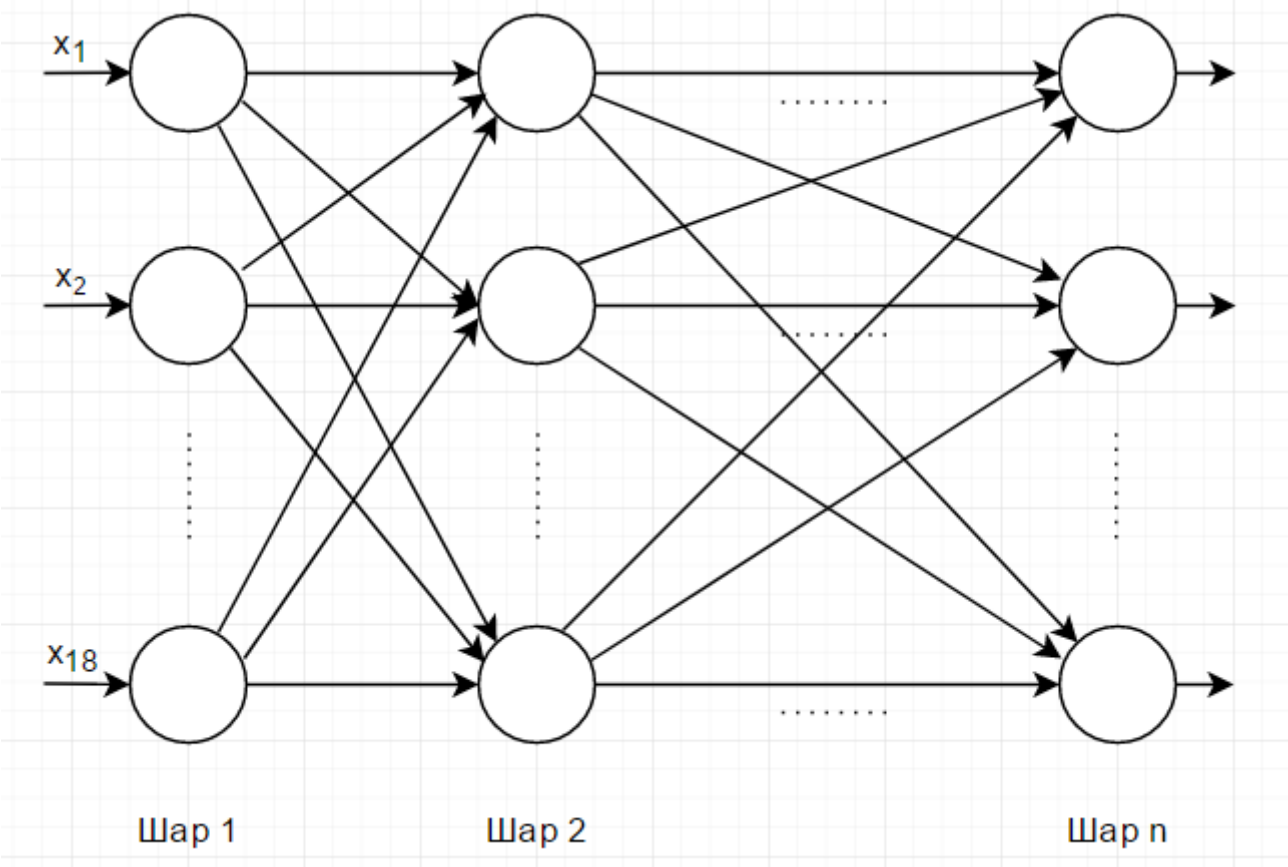


Рисунок 2.6 – Багатошарова нейронна мережа на 18 вхідних нейронів

Усі три моделі багатошарових персептронів досліджувались щодо виконання завдань цифрової обробки інформації, які виконуються в бортовими системами космічних апаратів.

За результатами досліджень виявилось, що із задачею оцінки технічного стану імітаційних моделей бортових підсистем космічних апаратів, впоралась лише третя модель багатошарового персептрона. Досліджувались бортові підсистеми космічних апаратів, що видають у якості телеметричної інформації 32-розрядні послідовності в бортових комплексах управління. Апаратна реалізація імітаційної моделі відбувалась за використання плати Xilinx ML605 одночасно з використанням архітектури багатошарових персептронів. Досліджуючи систему терморегулювання космічних апаратів, яка є однією з суміжних бортових систем. В результаті дослідження виявилось, що третя модель дозволила виявити та провести класифікацію двох типів помилок, які пов'язані як з програмними так і апаратними несправностями цієї системи.

Саме за допомогою третьої моделі багатошарового персептрона відбулося розпізнавання даних (векторів), що ведуть до «пропажі» контрольних сум на контролера шини мультиплексного каналу обміну та призводять до збільшення на 20 мс інтервалів між часом приходу контрольних послідовностей і часом їх реєстрації в системою контролю і діагностики зі складу бортового комплексу управління космічним апаратом.

Метою експерименту було навчання персептрона розпізнаванню матриці.

В той же час дві інші моделі багатошарових персептронів знизили рівень помилок щодо розпізнавання матриці обнулення сигнатур приблизно на 60 ... 70% виконавши близько 530 ітерацій, і витраченим на це часом близько 80 мс .

Використовуючи третю модель досягнення повного зниження рівня помилок у розпізнаванні матриці відбулося приблизно за 400 ітерацій, з витраченим часом до 60 мс.

Обмеженість ресурсів в даному випадку не виникала, проте мало місце зменшення часу та кількості ітерацій, в той же час у попередньому прикладі лише час обчислення персептрона був близький до 800 нс..

2.5 Використання радіально-базисних мереж для перетворення частотно-часових параметрів сигналів у код двох змінних

Розглянемо використання радіально-базисної нейронної мережі та, в той же час, проаналізуємо варіанти апаратної реалізації (рисунок 2.7) [14].

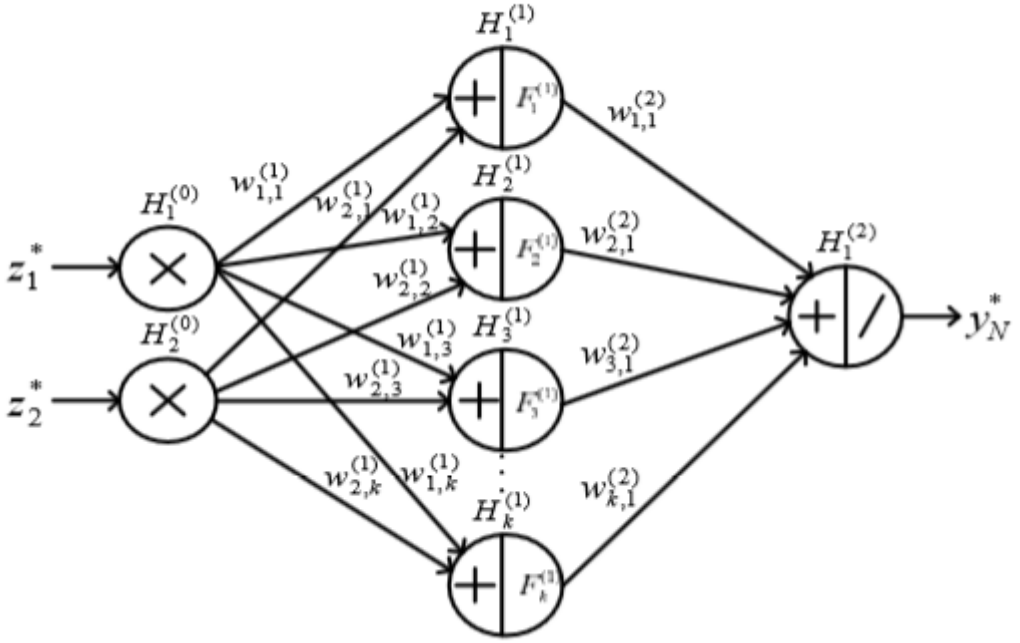


Рисунок 2.7 – Структура перетворювача на основі радіально-базисної нейронної мережі [14]

Обчислення експоненціальної функції на ПЛІС може бути проведено наступними способами:

а) розкладення через ряд Тейлора;

б) таблично.

Відносна похибка залежить від кількості членів ряду Тейлора. При використанні 7 доданків похибка не перевищує 1%, а при збільшенні кількості складових – прямує до нуля.

Похибка відтворення нелінійної залежності двох змінних залежить від кількості радіально-базисних нейронів. При використанні п’яти нейронів прихованого шару похибка складала близько 2%, а при використанні десяти нейронів – 0.2%.

2.6 Використання радіально-базисної нейронної мережі, яка реалізована на масивно-паралельній архітектурі графічного процесора

RBF-мережа являє собою двошарову мережу (рисунок 2.8).

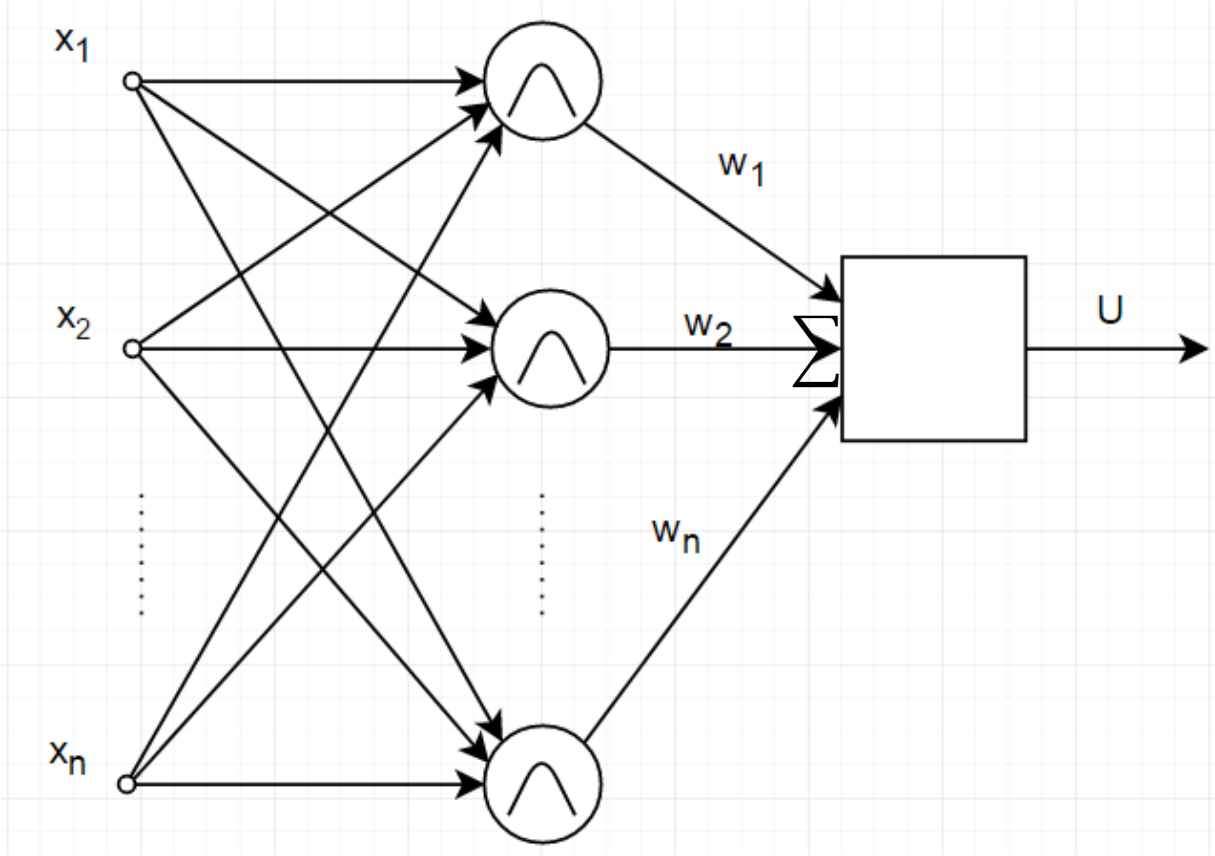


Рисунок 2.8 – Радіально-базисна нейронна мережа

Визначимо основні кроки навчання радіально-базисної мережі:

а) генерація початкових параметрів мережі;

б) генерація випадкових контрольних точок для одного або декількох циклів навчання;

в) декілька циклів налаштовування центрів і ширини при зафіксованих вагах;

г) декілька циклів налаштування ваг при сталих значеннях центрів і ширини;

д) перевірка досягнення необхідної похибки, якщо не досягнута, то повернення до кроку 2.

В даному прикладі використовувалось розпаралелювання обчислень на графічному інтерфейсі технології CUDA. Розпаралелювання призвело до зменшення час обчислень [15].

При розгляді даного прикладу обчислювалась швидкість на двох процесорах, як з використанням розпаралелювання на 64 нейронах так і без розпаралелювання обчислень. За результатами обчислень виявилось прискорення у 34.5 рази при роботі з 524 контрольними точками, в той же час, крок 3 без прискорення виконувався 71.7 мс, а з прискоренням – 0.58 мс.

Не наводимо час повної ітерації, тому що метою було дослідження прискорення навчання, а це мало місце лише при налаштовуванні центрів і ширини та при налаштуванні ваг.

Це дослідження підтвердило ефективність використання розпаралелювання обчислень.

2.7 Використання згорткової нейронної мережі, яка реалізована на програмованій логічній інтегральній схемі

Розглянемо детальніше використання ПЛІС. У згортковій мережі послідовність застосування операцій така: згортка, підвибірка, нелінійна функція активації (ReLU – rectified linear unit). На рисунку 2.9 представлена нейронна мережа, на вхід якої подається кольорове зображення розміром 224x224x3 і відбувається згортка з 96 фільтрами розміром 11x11x3 з кроком 4. Результатом цієї згортки маємо набір карт ознак розміром 55x55x96. Такий набір карт ознак опрацьовує нелінійна функція ReLU, після цього виконується згортка з 256 фільтрами 5x5x48 (при розбитті нейронної мережі на 2 частини), а потім і виконується операція підвибірки і випрямлення ReLU. Продовжуючи, отримуємо останній шар, який містить 4096 ознак [16].

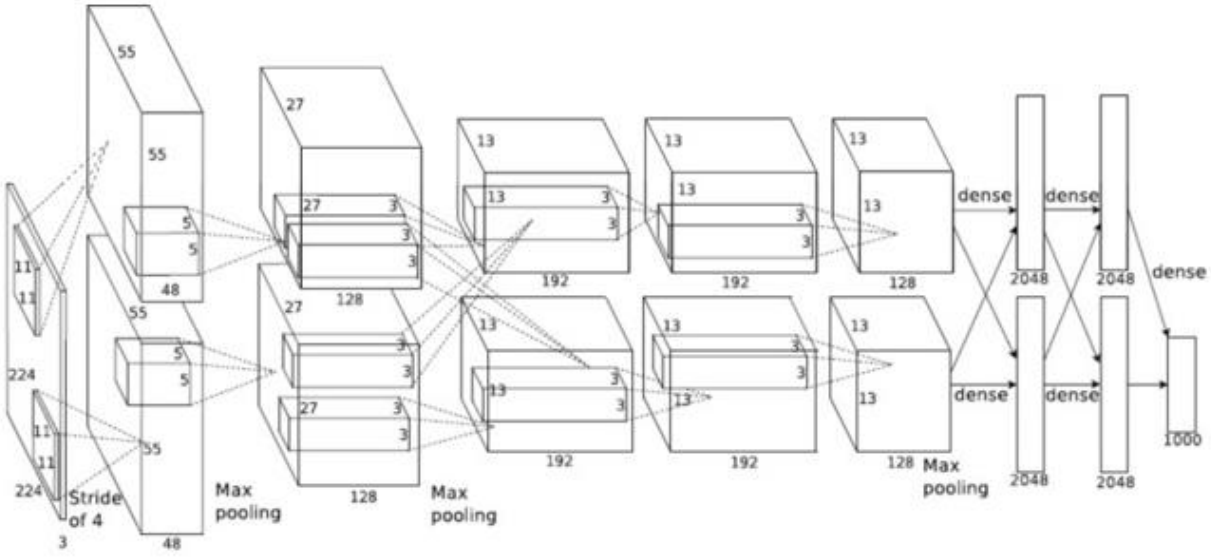


Рисунок 2.9 – Згорткова нейронна мережа для класифікації зображення[14]

Дані вхідного потоку подаються у вигляді беззнакових байт 0-255. ПЛІС містить елементи ядра: 16-бітові числа з фіксованою точкою і 12 бітами дробової частини. Реалізація операцій з плаваючою точкою на ПЛІС вимагає значних обчислювальних ресурсів, а в завданню згорткових мереж діапазон значень ядра є передбачуваним.

Підключена схема управління відбувається за допомогою шини AXI4-Lite. Дані завантажуються з основної пам’яті, а результат направляється через шину AXI4, використовуючи прямий доступу до пам’яті.

Реалізовані функції:

- завантаження ядер;
- обробка;
- заміна ядра;
- скидання.

Процес запису всіх ядер з використанням прямого доступу до пам’яті і є завантаженням ядер, які розміщені на ПЛІС.

Порядок оброблення даних:

а) отримання даних від джерела;

б) заповнення буфера FIFO для рядків;

в) обчислення згортки в кожній позиції;

г) оновлення та збереження значення для операції підвибірки (Maxpool);

д) збереження підсумкового значення (карти ознак) в локальній пам'яті;

е) зрушення буфера рядків.

У якості сховища для рядків зображення використовувалося 11 блоків пам’яті в режимі FIFO (рисунок 2.10). Ще один блок пам’яті використовувався для зберігання ядер. Блоки DSP48 використовувалися в якості арифметичного модуля. Ці блоки забезпечували фіксовану затримку тривалістю 1 такт щодо операцій множення і додавання.

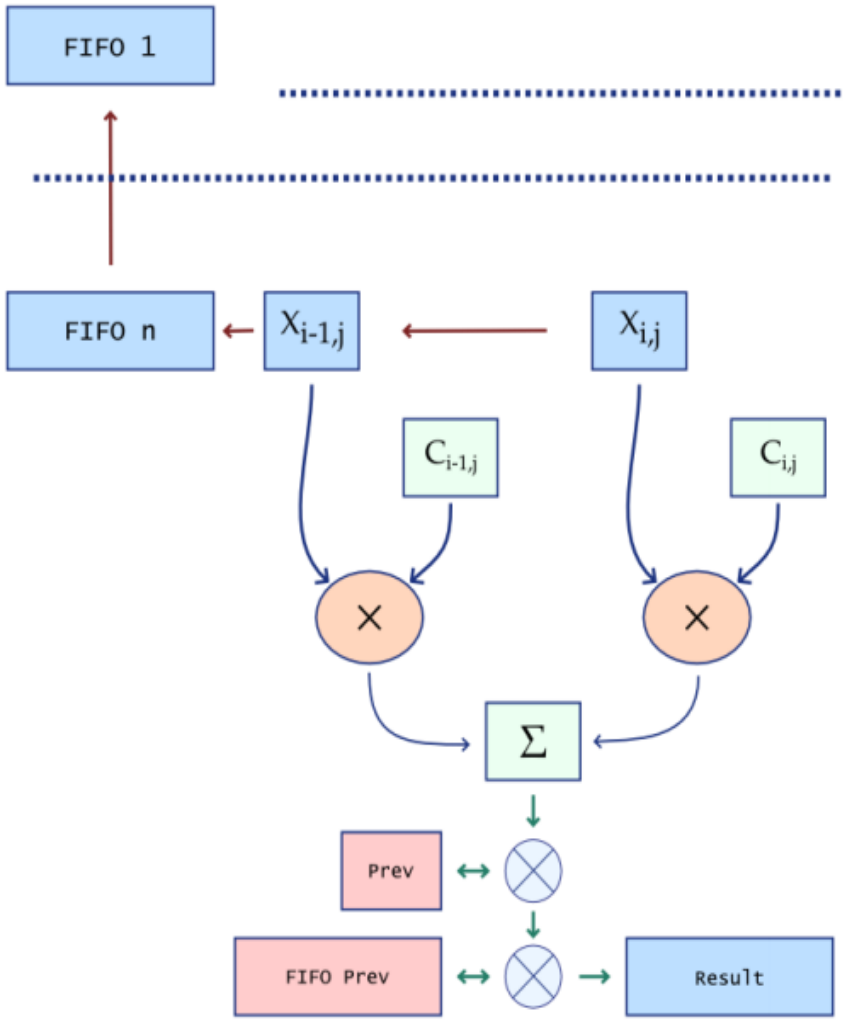


Рисунок 2.10 – Процес обробки сигналу

Процес згортки здійснювався за допомогою циклічного списку черг FIFO 1 – FIFO n безпосередньо з блоків пам’яті і набору регістрів. В результаті, інтенсивність кожного наступного пікселя зображення зсувалася послідовно через регістри останнього рядка, а потім записувалася в чергу FIFO n. В той же час на кожному такті виконувалося множення на коефіцієнт ядра. Такий процес еквівалентний горизонтальному руху ядра згортки.

Після досягнення кінця рядка, адреси лічильників FIFO змінювалися, тобто відбувався зсув ядра згортки на один піксель вниз.

Щодо операції підвибірки то результат згортки в непарних стовбцях непарних рядків записувався прямо в комірку попереднього значення Prev. У парних стовбцях непарних рядків результат згортки порівнювався з коміркою Prev і більше значення записувалося в блок пам’яті FIFO Prev. Для наступного (парного) рядка, послідовність дій була зворотною. Для непарного стовпця значення з пам’яті FIFO Prev зчитувалося і порівнювалося зі значенням згортки в даній точці, а вже потім записувалося в Prev.

Для парного стовпця парного рядка, після порівняння з попереднім значенням, максимальне значення (таким чином, що є підвибірки по чотирьом коміркам) записувалося в пам’ять результату.

2.8 Порівняння програмованої логічної інтегральної схеми з іншими платформами

Для дослідження швидкодії та швидкості навчання порівняємо ПЛІС з іншими платформами. У якості альтернатив обрано:

- інтегральна схема для специфічного застосування(ASIC);
- графічний процесор(GPU).

При використанні ASIC треба чекати, поки виробник виготовить її, тоді як для ПЛІС достатньо лише створити програмне забезпечення, також ПЛІС можна перепрограмовувати, це безсумнівні плюси. Серед плюсів ASIC лише менший розмір та менша вартість при великих замовленнях

GPU, на відміну від ПЛІС, апаратно алгоритми не реалізовують, а займаються виконанням програми. Отже, графічному процесору потрібно діставати програмні інструкції з пам’яті, виконати їх, переслати назад в пам’ять.

На відміну від GPU, ПЛІС не витрачає часу на пересилання даних через загальну шину, має багато однотипних логічних блоків для виконання типових логічних операції, тому виграш по швидкості є значним. Також для роботи графічного процесора потрібен CPU, що робить таку конструкцію дорогою.

Проте, GPU є набагато ефективнішими за ПЛІС в обчисленнях із плаваючою комою, що надає йому переваг при обробці графіки та сигналів.

Переваги ПЛІС:

- немає операційних витрат часу на пересилання даних;
- менше споживання енергії;
- менша ціна.

2.9 Висновок

Проаналізувавши низку прикладів можна винести певні результати. Було розглянуто: певні задачі, для яких використовували штучні нейронні мережі, багатошарові нейронні мережі з описом кожного шару та впливу їх на час виконання, реалізація на різних процесорах з описом алгоритму обчислення.

Також було розглянуто реалізація радіально-базисної нейронної мережі та реалізація іншої нейронної мережі на ПЛІС [18].

Отже, на швидкодію нейронів, швидкість навчання нейронних, точність вихідних результатів, затрачений об’єм пам’яті мереж впливає велика кількість факторів:

а) обчислювальний пристрій;

б) кількість нейронів;

в) кількість шарів;

г) складність обчислень нейронів;

д) алгоритм обчислень.

А також сам розмір архітектури може збільшити час однієї ітерації, але у той же момент зменшити загальний час навчання мережі.

Все це показує, на скільки різноманітними можуть бути нейронні мережі.

3 АПАРАТНО-ПРОГРАМНА РЕАЛІЗАЦІЯ ШНМ

Розробка застосування на програмованій логічній інтегральній схемі основується на описі алгоритму мовою VHDL. Це високорівнева мова опису апаратури. Технологія розробки включає трансляцію опису мовою VHDL в опис, який знаходиться на рівні функціональних компонентів ПЛІС та їх логічних таблиць. Проте прості функції у ПЛІС зазвичай реалізуються як замовлені віртуальні модулі або окремим проектом. В них вказують розрядність даних та інколи внутрішню будову.

Апаратно-програмна реалізація компонентів нейронних систем керування повинна використовувати методи і алгоритми, які орієнтуються на реалізації в ПЛІС, тобто повинні адаптуватись до їх елементного базису. Не залежно від фірми виробника, ПЛІС має у своєму складі блоки постійного запам’ятовуючого пристрою, оперативного запам’ятовуючого пристрою, логічні таблиці, елементи суматорів, тригери. Інші всі пристрої реалізуються на їхній основі. Щоб конкретизувати алгоритми, методи і критерії їх оптимізації, розглядається ПЛІС з архітектурою фірми Xilinx. У інших виробників такі критерії, як затрати часу та апаратні затрати, можна перерахувати.

Серед сучасних розробок, які використовують ПЛІС із високою інтеграцією відзначено у першу чергу виконаних на ПЛІС високої інтеграції «нейрочип-8», який зображено на рисунку 3.1. Це інструментальна плата XDSP-680 на базі ПЛІС із сімейства Spartan, яка належить компанії Xilinx. Нейромережева прошивка, ПЛІС сімейства Spartan займається Науковий центр нейрокомп'ютерів у Москві та «Скан Інжиніринг Телеком» із Воронежу. Ці два партнери також розробляють інші перспективні вироби: «нейрочип-2000» на базі ПЛІС сімейства Virtex/Virtex-E, інші інструментальні плати на базі ПЛІС різних серій, мезонінні модулі різного призначення, які дають змогу ефективно та швидко створити обчислювальні системи з різним функціональним призначенням. Зазначені вище розробки засновані на нейронах, які мають порогову функцію активації і призначаються для обчислювальної системи загального призначення. Серед сучасних нейроконтролерів

інтерес представляють ті, які здатні адаптуватись та функціонувати у реальному часі.

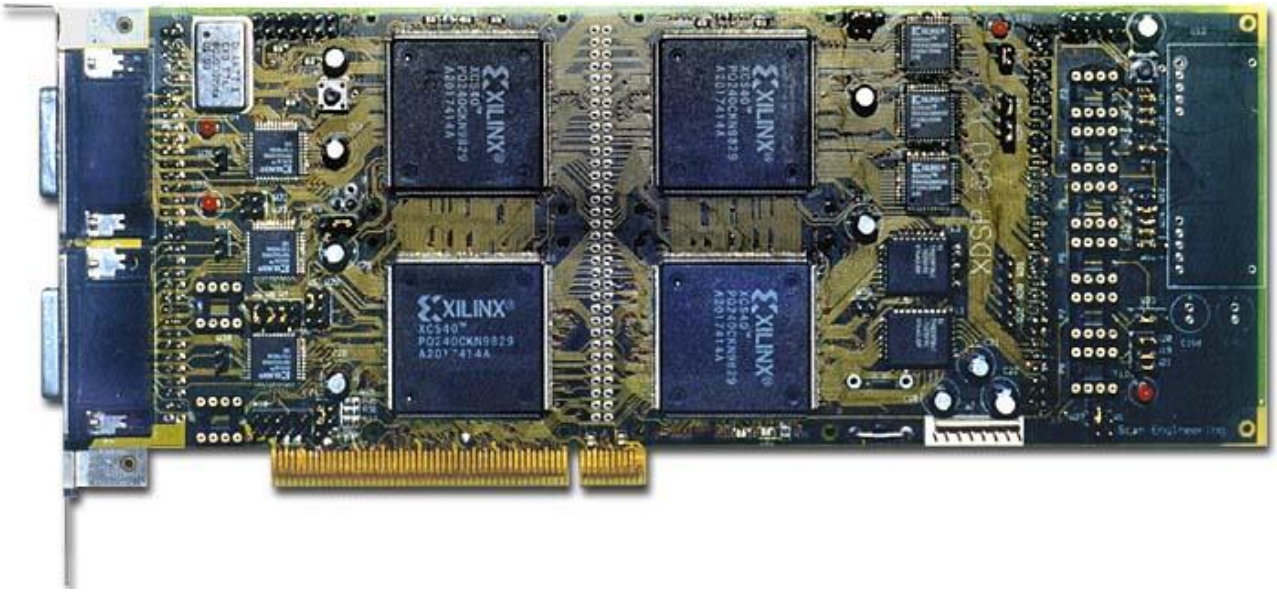


Рисунок 3.1 – Інструментальна плата XDSP-680 [19]

Проблема апаратно-програмної реалізації нейромережевої систем управління полягає у реалізації самого штучного нейрона та його функцій активації засобами програмованої користувачем вентильної матриці (FPGA) [20; 21].

На рисунку 3.2 представлено функціональну схему моделі безперервного штучного нейрона. Модель роботи нейрона має такий алгоритм. Сигнали $a_{k,i}$ надходять на вхід, де реалізується функція синапсів, де кожен має свій вагових коефіцієнт w_{ki} (синаптична вага). Після цього сигнали стають зваженими і надходять до лінійного суматора. Після додавання сигналів переходить до блоку активаційної функції f (S_k). Потім відбувається відповідна обробка і тоді параметр подається на вихід у вигляді сигналу q_k. Зазвичай функція активації обмежує сигнал нейрона у певному діапазоні, [0; 1] або [-1; 1]. Також сигнал нейрону змінюється від початкового зрушення b_k, який подається на вхідний сигнал блоку функції активації [22].

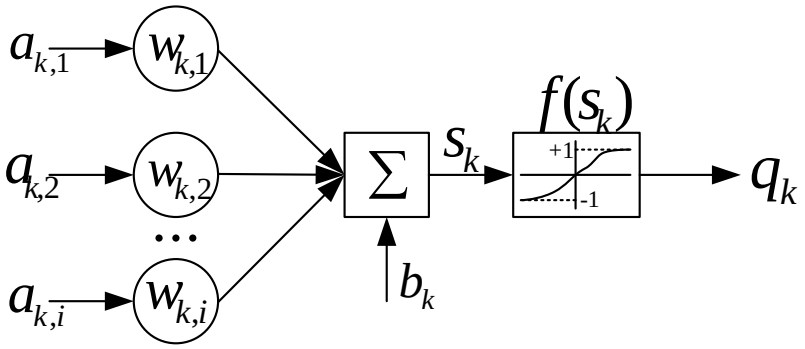


Рисунок 3.2 – Функціональна схема штучного нейрона

Наступними залежностями можна математично описати модель нейрона:

$$q_k = f\left(S_k\right) = f\left(\sum_{i=1}^{n} w_{k,i} a_{k,i} + b_k\right) \tag{3.1}$$

де q_k – вихідний сигнал k-го нейрона;

$f\ (S_k)$ – активаційна функція нейрона;

$a_{k,i}$ – вхідні сигнали k-го нейрона;

w_{ki} – синаптична вага k-го нейрона;

b_k – зміщення k-го нейрона.

В нейроні функція активації $f\ (S_k)$ виконує нелінійне перетворення, яке здійснює нейрон. Є багато видів функцій активації, проте найчастіше використовують наступні чотири: лінійну функцію, граничну функцію, сигмоїдальну функцію та функцію Гауса. Активаційні функції штучних нейронів представленні на рисунку 3.3.

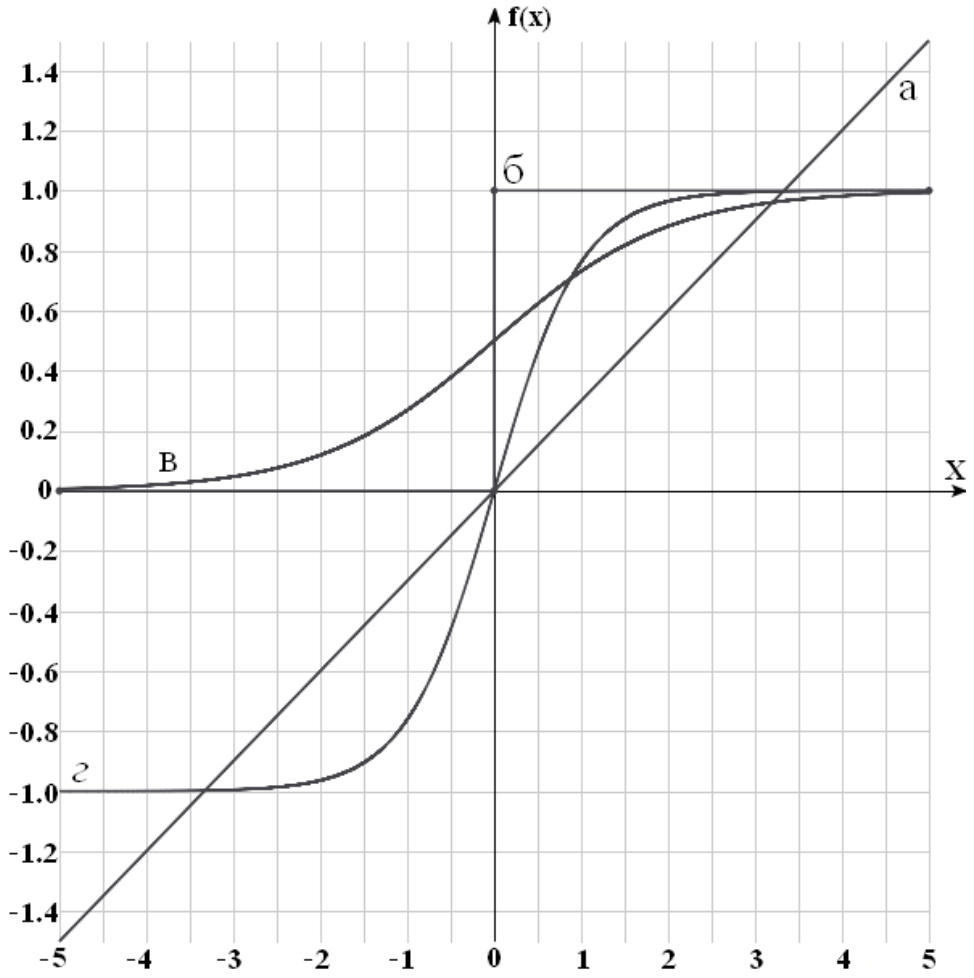


Рисунок 3.3 – Активаційні функції ШНМ

Лінійна функція активації (рисунок 3.3а) описується наступною залежністю:

$$f(x) = px. \tag{3.2}$$

Порогова функція (рисунок 3.3б) описується залежністю:

$$f(x) = \begin{cases} 1, \text{якщо } x \geq a \\ -1, \text{якщо } x < -a \end{cases}, \tag{3.3}$$

На рисунку 3.3б представлено вид кусково-лінійної передавальна функцыя при а=1.

Сигмоїдальна функція (рисунок 3.3в) має таку залежність:

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.4}$$

де а – параметр, що визначає нахил лінійної ділянки.

При нескінченно великому значенні параметра а функція вироджується в порогову.

Найбільш розповсюдженим типом функцій активації є сигмоїдальна функція. Вони безперервні, монотонно зростаючі і диференційовані. Саме диференційованість являється корисною властивістю для використання певних методів аналізу та навчання. Також сигмо їдальні функції мають властивість універсальної апроксимації. Особливість нейронів із даною функцією активації полягає у тому, що ця функція посилює сильні сигнали набагато менше за слабкі, бо області на сильних сигналах знаходяться на пологих ділянках характеристики. Це практично обрізає великі сигнали та запобігає їх насиченню.

Також при обчисленні методом зворотного поширення помилки сигмоїдальна функція активації дозволила істотно скоротити складність обчислення, бо просто виражати її похідну через саму функцію. Це робить даний метод зручним у використанні.

Не менш важлива причина введення нелінійності виступає математично доведена можливість отримання якого завгодно точного наближення бажаної неперервної функції з багатьма змінними. Використовуючи для цього операції додавання і множення на число, лінійні функції, суперпозицію функцій, або на одну довільну нелінійну неперервну функцію одної змінної.

Говорячи про сигмоїдальні функції, мається на увазі клас функцій, який описується виразом:

$$f(x,k,b,T,c) = k + \frac{c}{1+be^{Tk}}, \tag{3.5}$$

де x, k, b, T, c – параметри $k \in R$;

$b \in R, b > 0$;

$T, c \in R \setminus \{0\}$.

Коли прийняти k=0, b=1, c=1 і T=-1, то рівняння (3.5) матиме вигляд:

$$f(x,0,1,-1,1) = 0 + \frac{1}{1+1e^{-1x}} = \frac{1}{1+e^{-x}}. \quad (3.6)$$

Функція, яка описана формулою (3.6) називають «класичним» сигмоїдом (рисунок 3.2в).

Якщо прийняти k=1, b=1, c=-2 і T=2, тоді вираз (3.6) матиме вигляд:

$$f(x,1,1,2,-2) = 1 + \frac{-2}{1+1e^{2x}} = \frac{1+e^{2x}-2}{1+e^{2x}} = \frac{e^{2x}-1}{e^{2x}+1} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}. \quad (3.7)$$

Функція, яка описана виразом (3.7) називають гіперболічним тангенсом (рисунок 3.3г).

Область значень параметрів на вході знаходяться під активною областю визначення функцій активації нейрона. У ній і відбувається істотна зміна значень активаційної функції.

Для сигмоїдальної функції активації було використано інтервал [–4; 4], це область активного визначення, у цьому проміжку функція може досягати значень у інтервалі (0.017; 0.983), що відповідає 96.5 % від усієї області значень. Для сигмоїдальної функції, радіально-базисної функцій та функції з гіперболічним тангенсом пропонуємо брати запропоновано брати інтервал [–2; 2] як активну область визначення. У цих значеннях функції набувають значень в інтервалах (–0.965; 0,965) та (0.0183; 1], відповідно.

Зі сторони цифрової реалізації найскладнішим є реалізація нелінійних функцій активації.

Коли розробляють метод та алгоритми реалізації нейронів з сигмоїдальної активаційною функцією, варто пам’ятати про особливі будови ПЛІС, бо в ній є суматори, мультиплекс ори, ресурси логічних таблиць, проте немає блоків поділу. Нейрон повинен бути таким, щоб можна було регулювати точність обчислень та розрядність даних на вхід.

Існують різні підходи цифрової реалізації нелінійних активаційних функцій. Зазвичай використовуються різні способи апроксимації, такі як розкладання у ряд Тейлора, кусково-лінійна апроксимація або табличний метод. Розкладати у ряд Тейлора немає сенсу, адже цей метод вимагає використання багатьох математичних операцій. В ПЛІС такий метод неприйнятний для реалізації, бо у розкладанні у ряд Тейлора використовується множення, а цей блок займає велику кількість обчислювального ресурсу. При використанні табличного методу створюються глобальні змінні та таблиці з можливими значеннями цільових функцій, а також забезпечується неконтрольований та непередбачуваний доступ до неї усіма нейронами мережі. Це відповідно створює велику часову затримку, а створювати окрему локальну таблицю для кожного нейрона мережі не ефективно зі сторони використання обчислювального ресурсу ПЛІС [23]. Тому кусково-лінійна апроксимація прийнята найоптимальнішим методом реалізації активаційних функцій сигмоїдального типу.

Пропонується наступний метод реалізації штучного нейрону з активаційними функціями сигмоїдального типу.

На першому кроці виконується дослідження функції активації на симетрію відносно осей. Це представлено на функції (3.8).

$$f(-x)=1-f(x)=1-\frac{1}{1+e^{-x}}=1-\frac{1}{1+\frac{1}{e^{x}}}=1-\frac{e^{x}}{e^{x}+1}=\frac{e^{x}+1}{e^{x}+1}-\frac{e^{x}}{e^{x}+1}=\frac{1}{e^{x}+1}, \qquad (3.8)$$

отримаємо:

$$1-\frac{1}{1+e^{-x}}=\frac{1}{1+e^{x}},\quad \text{або}\quad 1-f(x)=f(-x). \tag{3.9}$$

Розглянемо $f(x)$ лише для додатних аргументів. А для його від'ємні значення можна знайти з формули (3.9). Це у свою чергу пришвидшить обчислення значення функції та зменшить ресурс ПЛІС.

На другому кроці задається кусково-лінійна функцію на кожному інтервалі $(-\infty;x_1),(x_1;x_2);...(x_n;+\infty)$ окремою формулою. Записуємо це у вигляді системи рівнянь:

$$f(x)=\begin{cases} k_0x+b_0,\ x<x_1 \\ k_1x+b_1,\ x_1<x<x_2 \\ \cdots \\ k_nx+b_n,\ x_n<x \end{cases}, \tag{3.10}$$

На третьому кроці розраховується значення *f(x)*, для уже розрахованого з пам'яті значення *x*, а також проводиться відбір коефіцієнтів *k* та *b* з формули (3.10), для від'ємних аргументів, значення функції вираховуються формулою (3.9).

Отже, на основі лінійних функцій для якого-завгодно аргументу з бажаною точністю можна апроксимувати будь-яку нелінійну функцію таким же чином, як і сигмоїдальну.

4 АЛГОРИТМ АПАРАТНОЇ РЕАЛІЗАЦІЇ ШТУЧНОГО НЕЙРОНА З СИГМОЇДАЛЬНОЮ ФУНКЦІЄЮ АКТИВАЦІЇ

У даному розділі розглянуто алгоритм апаратної реалізації штучного нейрона з функцією активації сигмоїдального типу на ПЛІС. Для реалізації використовується мова VHDL [27].

На першому кроці задаються коефіцієнти штучного нейрона.

Кожен нейрон отримує під контроль відповідний блок оперативної пам'яті, де потім зберігаються вагові коефіцієнти. Для задання ваг кожного нейрона використовуються наступні сигнали:

- synaddr (synapse address) – вибирається синапсис, вага котрого буде записуватися або зчитуватися за його номером у бінарному коді. Для нейрона з 4-ма синапсисами цей вхід буде 2-х бітним, з 8 – 3-х і т.п.;
- synsetw (synapse set weight) – тут зберігається значення коефіцієнту, яке треба записати в синапсис;
- synwren (synapse write enabled) – коли цей сигнал дорівнює 1 на вході якого нейрон записує в обраний синапс значення з synsetw, а коли цей сигнал дорівнює 0, тоді нічого не відбувається.

Під час кроку 2 на входи штучного нейрона подається значення вхідного вектора, після чого задається змінну *x*, яка дорівнює виходу суматора:

$$x = \sum_{i=1}^{N} w_i \cdot a_i \tag{4.1}$$

де a – входи нейрона;

w – вагові коефіцієнти нейрона.

Для обчислення використовуються числа із фіксованою точкою. Кожне число має в собі 16 біт, з них 9 приходиться на цілу частину, а 7 – на дробову. Для використання арифметичних операцій над даними числами використовують

бібліотеку fixed_pkg_c.vhdl. Ця бібліотека синтезується, її арифметичні операції добре оптимізовані.

Крок 3. Розраховується модуль від аргументу сигмоїдальної функції, для цього задаємо змінну x':

$$x' = |x| \tag{4.2}$$

На четвертому кроці виконується розбивання сигмоїдальної функції активації на лінійні шматки та визначаються коефіцієнти лінійних рівнянь k і b. Розбивання на лінійні проміжки представлено на рисунку 4.1.

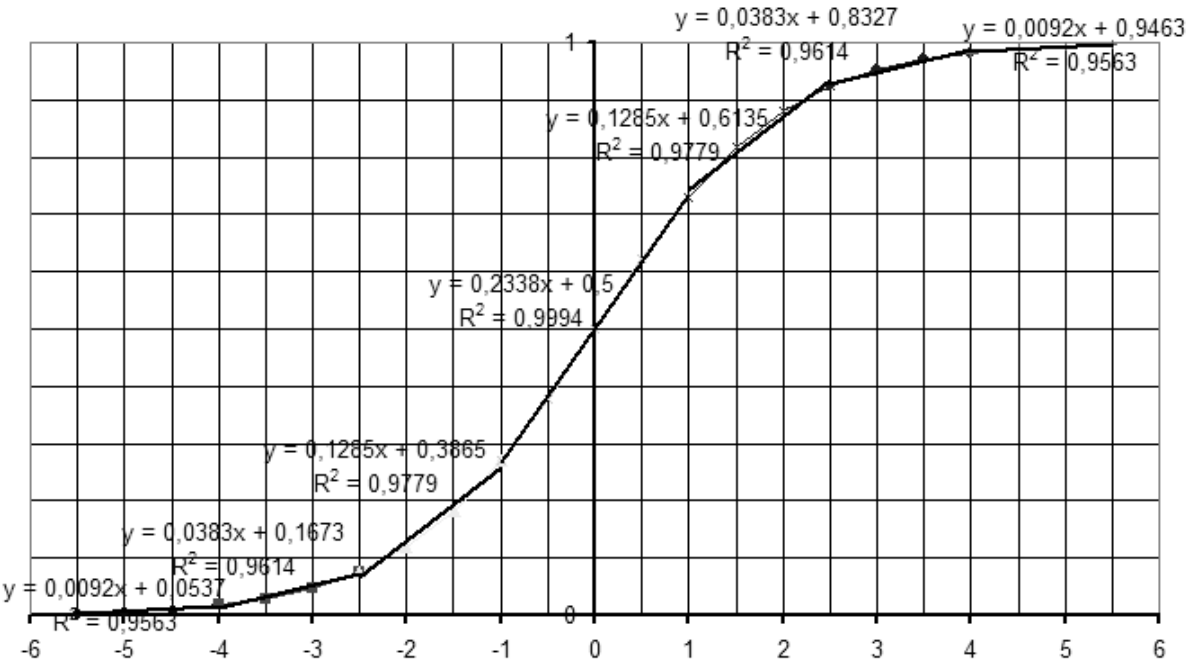


Рисунок 4.1 – Кусково-лінійна апроксимація сигмоїдальної функції

Далі визначаються коефіцієнти лінійних рівнянь:

$$k, b = \begin{cases} [0.234, 0.500], \text{якщо } 0 \le x' < 1 \\ [0.129, 0.605], \text{якщо } 1 \le x' < 2.5 \\ [0.038, 0.833], \text{якщо } 2.5 \le x' < 4 \\ [0.009, 0.946], \text{якщо } x' \ge 4 \end{cases} \tag{4.3}$$

Крок 5. Визначається змінна *f*, значення обчислюється за формулою:

$$f = k \cdot x' + b \tag{4.4}$$

Крок 6. При умові, якщо $x<0$, значення локальної змінної обраховується за формулою:

$$f = 1 - f \tag{4.5}$$

На кроці 7 задається вихідний сигнал нейрона, який дорівнює значенню змінної *f*.

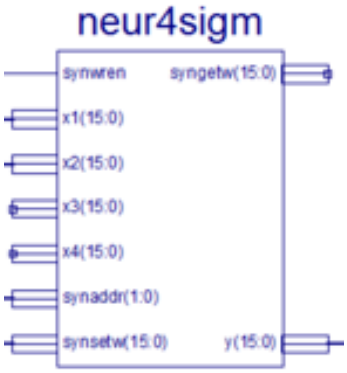


Рисунок 4.2 – Блок штучного нейрона в ПЛІС

За даним алгоритмом була реалізована на ПЛІС штучна нейронна мережа, яка містить чотири входи та нейрон з функцією активації сигмоїдального типу. Використовувались 16-розрядні числа із фіксованою точкою, така мережа зайняла 672 вентилів логічної матриці LUTs (Look Up Table). Реалізований на ПЛІС штучний нейрон є окремим обчислювальним блоком який представлено на рисунку 4.2. Абсолютна похибка складає не більше 0.005. Точність даної реалізації сигмоїдальної функції представлена на рисунку 4.3. Швидкодію можна виразити сумарною затримкою комбінаційної схеми цього блоку нейрона. Час, який витратився на одне проходження нейрона склав 75.6 нс. Якщо зменшувати або

збільшувати кількість лінійних відрізків, кількість вхідних векторів нейрона або розрядність чисел, буде також змінюватись для одного нейрона використаний ресурс ПЛІС, точність та швидкодія.

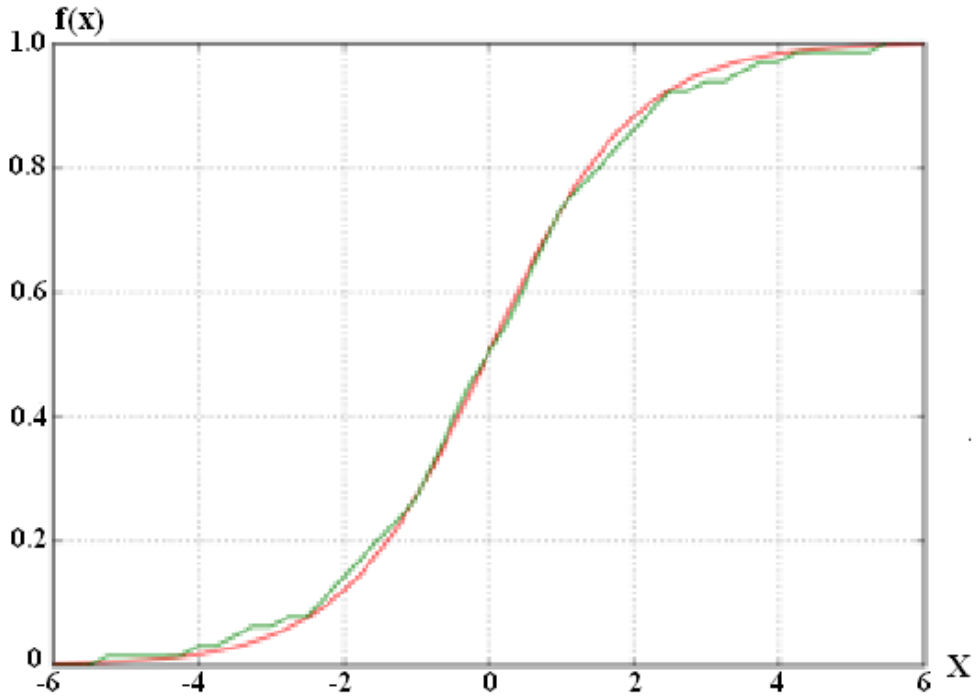


Рисунок 4.3 – Графік сигмоїдальної функції, реалізованої в ПЛІС

На рисунку 4.3 представлено класичну сигмоїдальну функцію за формулою 3.6 та функцію реалізовану в ПЛІС за розробленим алгоритмом. Як видно реалізований обчислювальний блок сигмоїдальної функції в ПЛІС досить точно відображає її, для подальшої реалізації штучних нейронних мереж та компонентів нейромережевих систем управління на їх базі.

Однією з головних особливостей нейромереж є паралельна обробка сигналів. Багатошарові нейронні мережі представляють собою однорідну обчислювальну середу. По термінології нейроінформатики це універсальні паралельні обчислювальні структури, призначенні для вирішення самих різних класів задач. Розглянемо основні принципи паралельної обробки сигналів на ПЛІС та концепцію реалізації нейронних мереж на ПЛІС.

Обробка за допомогою конвеєра – це метод, при якому кілька інструкцій або пакетів даних можуть оброблятися процесором паралельно. Для цього вся процедура обробки однієї команди розбивається на кілька етапів, по завершенні кожного з яких результат виконання зберігається в тимчасових регістрах, за допомогою яких окремі етапи і комутують між собою.

Розгортання внутрішніх циклів є відомим методом, який у ряді випадків може привести до значного зростання продуктивності. Суть методу полягає в поданні циклічних алгоритмів обробки з кінцевим числом ітерацій у вигляді довгої комбінаційної ланцюжка, що обробляється за один такт.

При багатопроцесорній обробці більше ніж один процесор в системі робить обробку вступників команд, тим самим дозволяючи здійснювати повністю паралельну обробку.

При програмно-апаратній реалізації штучних нейронних мереж на ПЛІС, кожний шар мережі працює паралельно іншому, тут використаний принцип конвеєра. Нейрони в кожному шарі також працюють паралельно за принципом багатопроцесорної обробки даних. Тобто кожний штучний нейрон мережі є окремим процесором і обробка інформації в кожному нейроні проходить одночасно.

Кожний нейрон представляється окремим блоком, як показано на рисунку 4.4, який в свою чергу складається з декількох паралельних процесів, а нейронна мережа багатопроцесорною системою. Мова програмування дозволяє явно вказувати сигнали, які запускають процес. Для запуску нейрона використовується вхідний сигнал цього нейрона.

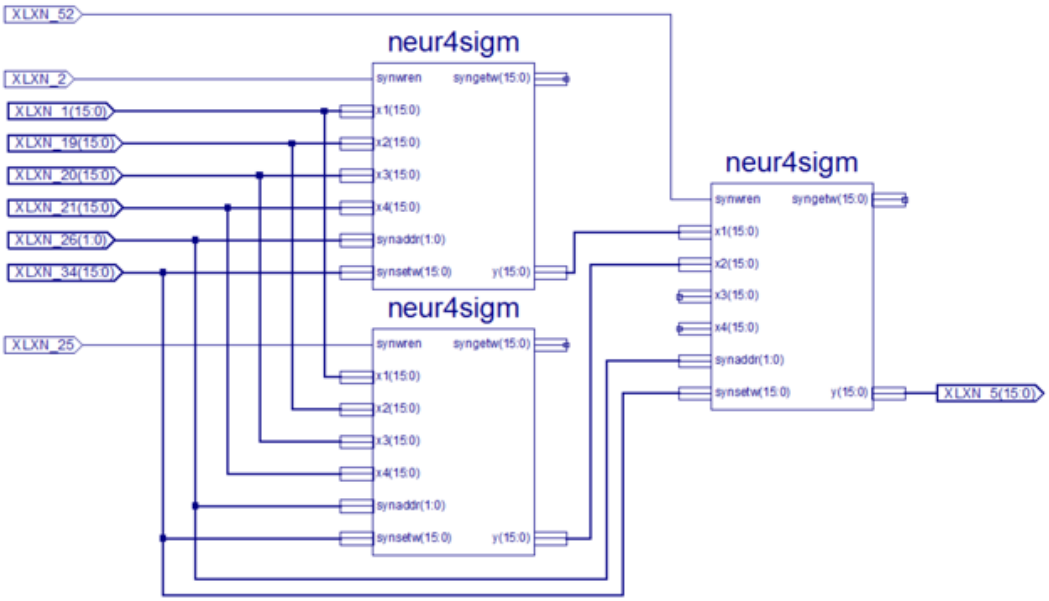


Рисунок 4.4 – Нейронна мережа побудована на ПЛІС

5 АЛГОРИТМ АПАРАТНОЇ РЕАЛІЗАЦІЇ НЕЙРОНІВ ПРИХОВАНОГО ШАРУ МЕРЕЖІ РАДІАЛЬНО БАЗИСНИХ ФУНКЦІЙ

Як зазначалось вище, мережі радіально базисних функцій, або ще RBF-мережі, це мережі які використовують радіально-базисні функції у якості активаційних функцій.

По своїй структурі RBF-мережі відносять до двошарових нейронних мереж, де існує прихований шар. Він має фіксовані вагові коефіцієнти та використовується для нелінійного перетворення вектора входу. Даний шар статично відображує вхідні змінні $\mathbf{r} \in R^n$ у нові $\mathbf{q}_1^{(l)} = col(q_1^{(l)},..., q_m^{(l)})$. Другий шар вихідний, на вхід він отримує дані з виходів прихованого шару та множить їх з коефіцієнтами, які налагоджуються $w_i = col(w_{i,1}, w_{i,2}..., w_{i,m})$. Тоді можна записати і-й скалярний вихід RBF-мережі у такому вигляді:

$$q_i^{(2)} = q_i = F(\mathbf{r}) = \sum_{j=1}^{m} w_{i,j} f_i(\mathbf{r}) + w_0 \cdot 1, \quad i = 1,2,...,K \tag{5.1}$$

Структурна схема RBF-мережі з вхідним вектором $\mathbf{r} \in R^n$ і вихідним вектором $\mathbf{q}^{(2)} = \mathbf{q} \in R^n$ представлена на рисунку 5.1 (додаток В). Активаційні функції вибираються серед симетричних радіальних функцій: [17]

$$f_i(\mathbf{r}) = f(\|\mathbf{r} - \mathbf{a}_j^*\|; h), \quad \mathbf{a}_j^* \in R^n. \tag{5.2}$$

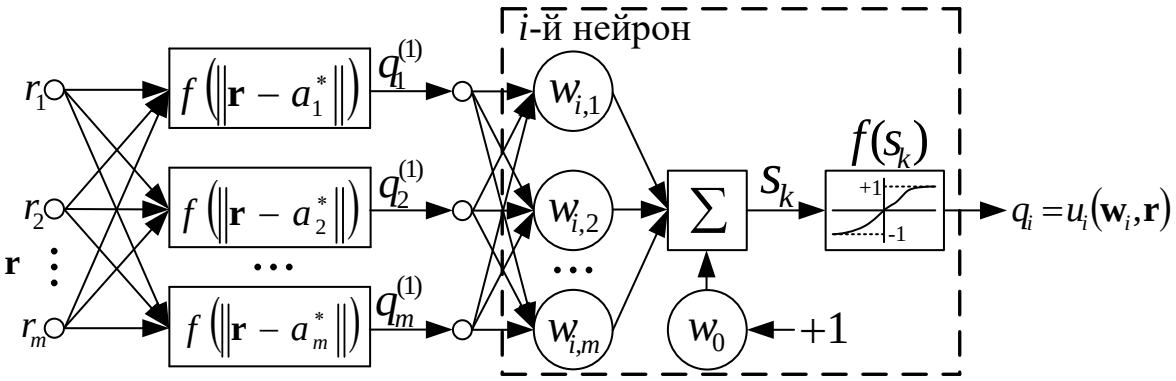


Рисунок 5.1 – Структурна схема RBF-мережі

Функції мають екстремуми при зміні вхідних сигналів *r* тільки поруч встановлених значень ваг нейронів у прихованому шарі $\mathbf{w}_j^{(1)} = \mathbf{a}_j^*,\ j = 1,2,...,m;$.

У схемі видно вагові коефіцієнти на виході прихованого шару $w_{i,j} = w_{i,j}^{(2)}$. Параметр *h* відповідає за положення центру поля сприйнятливості, який залежить від встановлених значень коефіцієнтів у першому шарі $\mathbf{w}_j^{(1)} = \mathbf{a}_j^*$. А форма функцій залежить від цієї величини $f_j(\cdot)$. Все це у сукупності визначає властивість RBF-мережі.

В якості активаційної функції *f(z)* може бути обрана одна з наступних функцій:

$$\exp(-z^2);\ \exp(-z);\ 1/(1+z)^2;\ 1/1+z^2;\ z^{\alpha}\log z;\ z^2+\alpha z+\beta.$$

Хоча на практиці найчастіше використовують першу із наведених функцій – функцію Гауса:

$$f_j(\mathbf{r}) = \exp\left(\sum_{k=1}^{n} 0.5\sigma_{j,k}^{-2}\left\|\mathbf{r}-\mathbf{a}_j^*\right\|^2\right) \qquad (5.3)$$

де $a_j^* = col(a_{j,1}^*, a_{j,2}^*, ..., a_{j,n}^*)$ – числовий вектор;

$\sigma_{j,k}$ – параметр, що відповідає за ширину функції Гауса.

Специфіка мережі радіально базисних функцій полягає у тому, що під час роботи в них відбувається налагодження лише вагових коефіцієнтів лінійного шару на вході системи. Помилка апроксимації обчислюється як у всіх багатошарових нейронних мережах безпосередньо на виході мереж, але налагодження тільки одного лінійного по параметрах налагодження шару знімає проблему пошуку глобального мінімуму функціонала навчання, зазвичай, квадратичного, і сприяє швидкому процесу навчання мережі.

Для апроксимації є проблематичною задачею вибір кількості радіально базисних функцій у RBF-мереж, ця проблема стає вирішальним чинником у використанні цих мереж у задачах управління та ідентифікації, де потрібна інформація для визначення розмірності нейронної мережі. Відомо, що кількість радіально базисних функцій зростає експоненціально зі зростанням кількості вхідних змінних. У нашому випадку число вхідних даних рівне *n*. Отже, RBF-мережі доцільно використовувати лише в тих задачах, де відома та розмірність вхідної множини

Як розглядалось вище, по своїй структурі RBF-мережі відносять до двошарових нейронних мереж, де існує прихований шар. Він має фіксовані вагові коефіцієнти та використовується для нелінійного перетворення вектора входу. Даний шар статично відображує вхідні змінні $\mathbf{r} \in R^n$ у нові $\mathbf{q}_1^{(l)} = col(q_1^{(l)},..., q_m^{(l)})$. Другий шар вихідний, на вхід він отримує дані з виходів прихованого шару та множить їх з коефіцієнтами, які налагоджуються за даною формулою $w_i = col(w_{i,1}, w_{i,2}..., w_{i,m})$.

Одною з основних проблем апаратно-програмної реалізації RBF-мереж є реалізація самого прихованого шару та його нелінійних активаційних функцій $f_i(r)$, у тому числі реалізація функції Гауса.

Функція Гауса представлена таким виразом (рисунок 5.2):

$$y = ae^{-\frac{(x-b)^2}{2\sigma^2}} \tag{5.4}$$

де a відповідає за висоту функції;

b – за центр найвищого значення на осі x;

σ – за різкість спадання функції в залежності від відстані вектора до центру.

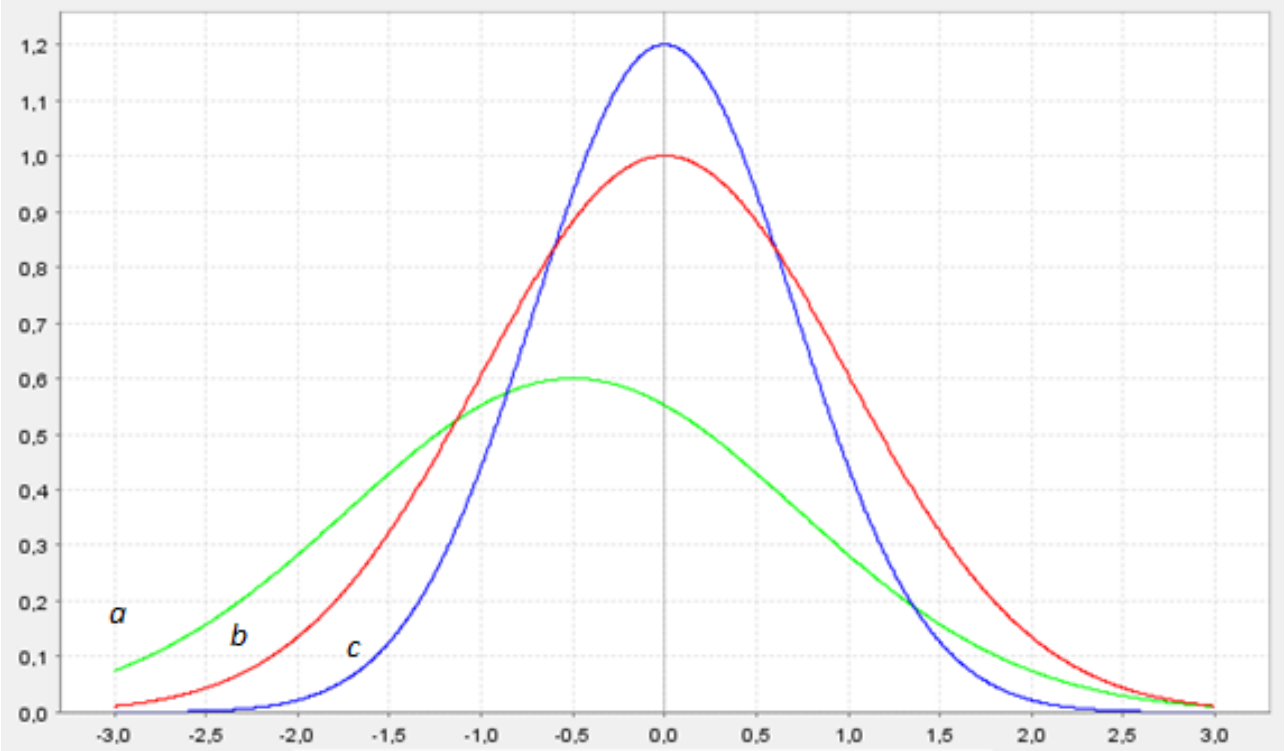


Рисунок 5.2 – Графіки функцій Гауса

Графіки функцій Гауса представлено на рисунку 5.2. На рисунку 5.2a представлено графік, що має пологе спадання та зсув по осі Ox. Ця функція описується таким чином:

$$y = 0.6e^{-\frac{(x+0.5)^2}{3}} \tag{5.5}$$

На рисунку 5.2c зображена більш видовжена функція, яка описується наступним чином:

$$y = 1.2e^{-\frac{x^2}{1}} \tag{5.6}$$

Зсув та різкість спадання можуть підбиратися при навчанні нейрону у залежності від потрібного результату.

Далі буде описано процес реалізації функції наступного початкового вигляду функції Гауса (рисунок 5.2b):

$$y = e^{-\frac{x^2}{2}} \tag{5.7}$$

На першому етапі на входи нейрона подається значення вхідного вектора, задається 16-бітна змінна *x* (рисунок 5.3).

```
entity Gaussian_function is
   Port (x: in STD_LOGIC_VECTOR(15 downto 0);
         y: out STD_LOGIC_VECTOR(15 downto 0));
end Gaussian_function;
```

Рисунок 5.3 – Реалізація першого кроку

Потім ми конвертуємо вхідні дані до 16-ти бітного числа з фіксованою точкою, 9 біт приходить на цілу частину і 7 на дробову(рисунок 5.4).

```
absx := abs(to_sfixed(x, 9, -6));
```

Рисунок 5.4 – Реалізація другого кроку

Функція Гауса є парною, тому на третьому кроці треба знайти модуль від аргументу x', проте спершу віднімається зміщення відносно осі *Oy*:

$$x' = |x - r| \tag{5.8}$$

Крок 4. На цьому кроці виконується процес розбиття функції активації Гауса на лінійні куски, і визначення коефіцієнтів лінійних рівнянь k та b. Розбиття рівняння на лінійні куски представлено на рисунку 5.5 а також у додатку Ґ.

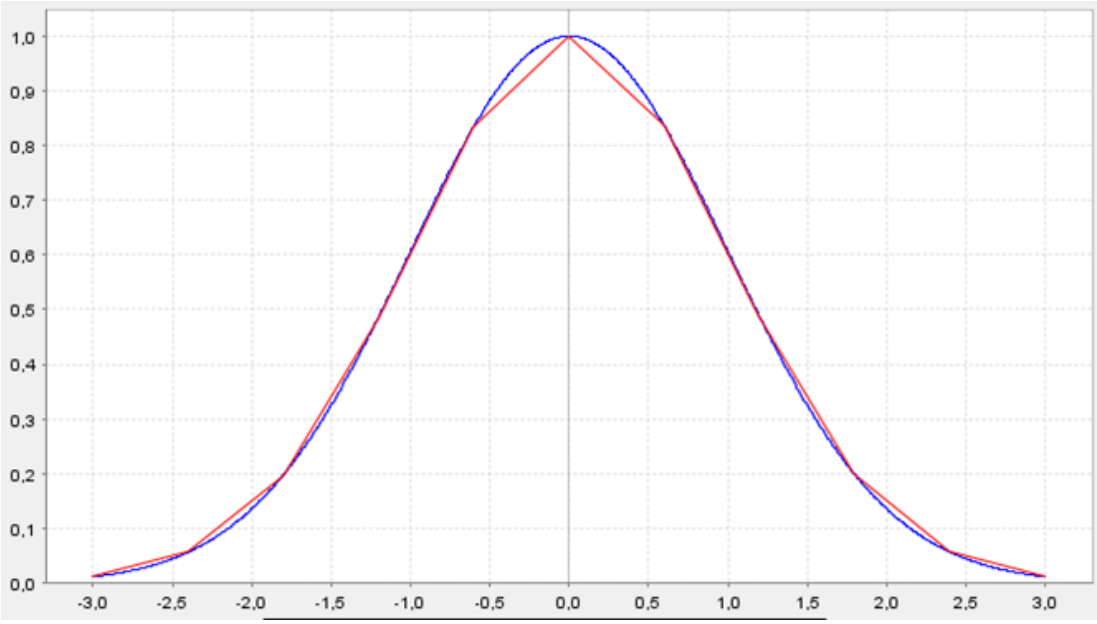


Рисунок 5.5 – Кусково-лінійна апроксимація функції Гауса

Визначаються коефіцієнти утворених лінійних рівнянь:

$$k, b = \begin{cases} [-0.275, 1], \text{якщо } 0 \leq x' < 0.6 \\ [-0.58, 1.183], \text{якщо } 0.6 \leq x' < 1.2 \\ [-0.48, 1.063], \text{якщо } 1.2 \leq x' < 1.8 \\ [-0.238, 0.628], \text{якщо } 1.8 \leq x' < 2.4 \\ [-0.093, 0.28], \text{якщо } x' \geq 2.4 \end{cases} \quad (5.9)$$

```
if (absx < to_sfixed(0.6, absx)) then
   k := to_sfixed(-0.275, k);
   b := to_sfixed(1, b);
elsif (absx < to_sfixed(1.2, absx)) then
   k := to_sfixed(-0.58, k);
   b := to_sfixed(1.183, b);
elsif (absx < to_sfixed(1.8, absx)) then
   k := to_sfixed(-0.48, k);
   b := to_sfixed(1.063, b);
elsif (absx < to_sfixed(2.4, absx)) then
   k := to_sfixed(-0.238, k);
   b := to_sfixed(0.628, b);
elsif (absx < to_sfixed(3, absx)) then
   k := to_sfixed(-0.093, k);
   b := to_sfixed(0.28, b);
else
   k := to_sfixed(0, k);
   b := to_sfixed(0, b);
end if;
```

Рисунок 5.6 – Реалізація кроку 4

На кроці 5 визначається змінна *f*, значення якої обчислюється за даною формулою:

$$f = k \cdot x' + b \tag{5.10}$$

Крок 6. Вихідна змінна *f* подається далі на нейрон із сигмоїдальною активаційною функцією, яка потім конвертується у змінну типу STD_LOGIC_VECTOR.

```
result := (r_kx + b);

y <= to_slv(resize(result, 15, 0));
```

Рисунок 5.7 – Реалізація кроків 5 і 6

На рисунку 5.8 представлений реалізований у середовищі розробки нейрон прихованого шару на ПЛІС. Повна реалізація функції Гауса представлена у додатку Б.

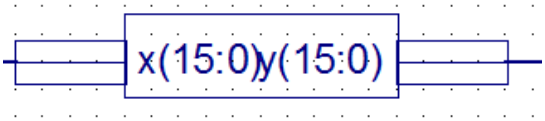


Рисунок 5.8 – Нейрон з функцією Гауса

Загальний вигляд мережа радіально базисних функцій із чотирма нейронами у прихованому шарі та одним нейроном із сигмоїдальною активаційною функцією представлено на рисунку 5.9.

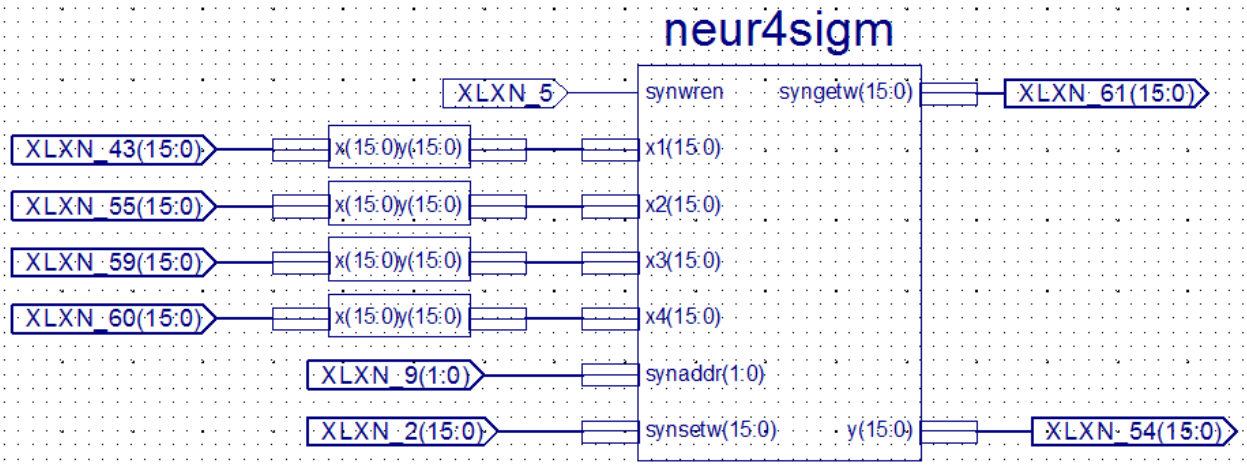


Рисунок 5.9 – RBF-мережа

За даним алгоритмом була реалізована на ПЛІС RBF-мережа, яка містить чотири нейрони прихованого шару та один нейрон з функцією активації сигмоїдального типу (додаток Д). Було використано 16-розрядні числа із фіксованою точкою, така мережа зайняла 1193 LUTs. Кожен нейрон прихованого шару мережі реалізований на ПЛІС у вигляді окремого обчислювального блоку, як видно на рисунку 5.7 та займає він 93 LUTs. Абсолютна похибка складає не більше 0.005. Побачити точність реалізації можна функції Гауса можна на рисунку 5.5. Загальний час обчислення реалізованої нейронної мережі зайняв 114.1 нс.

6. РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ МЕРЕЖІ РАДІАЛЬНО БАЗИСНИХ ФУНКЦІЙ

Штучний нейрон може бути зконфігурованний за наведеним вище алгоритмом у ПЛІС різних серій. В таблиці 2 показані результати такого конфігурування нейронної мережі, зображеної на рисунку 5.9.

Табл. 1. – Результати моделювання штучного нейрону в ПЛІС різних серій

ПЛІС		Швидкодія, нс	Ресурс ПЛІС (LUTs)
Серія	Speed		
Xilinx Spartan 3	-5	101.579	1193
	-4	117.805	
Xilinx Spartan 6	-2	6.916	423
	-3	6.032	
Xilinx Zinq	-3	0.96	420
	-2	1.102	
	-1	1.316	
Virtex 7	-1	1.316	420
	-2	1.102	
	-3	0.96	
Virtex 6	-2	1.209	420
	-1	1.376	
Virtex 5	-2	49.039	991
	-1	57.75	
Virtex 4	-10	70.517	1181
	-11	61.198	
	-12	53.903	
Kintex 7	-3	0.96	420
	-2	1.102	
	-1	1.316	
Artix 7	-1	1.588	420
	-2	1.345	
	-3	1.158	

Згадані метод та алгоритми апаратно-програмної реалізації нейрона та нейронних мереж дозволяють регулювати точність обчислень при різній розрядності вхідних даних, визначати займаний ресурс ПЛІС в кількості вентилів логічної матриці LUTs.

Однією з головних особливостей нейромереж є паралельна обробка сигналів. Багатошарові нейронні мережі представляють собою однорідне обчислювальне середовище. По термінології нейроінформатики це універсальні паралельні обчислювальні структури, призначенні для вирішення самих різних класів задач [10].

При апаратній реалізації штучних нейронних мереж на ПЛІС, кожний шар мережі працює паралельно іншому, що дозволяє при обчисленнях використати принцип конвеєра. Нейрони в кожному шарі також працюють паралельно за принципом багатопроцесорної обробки даних. Тобто кожний штучний нейрон мережі є окремим процесором і обробка інформації в кожному нейроні проходить одночасно.

Кожний нейрон представляється окремим блоком, як показано на рисунку 3, який в свою чергу складається з декількох паралельних процесів, а нейронна мережа є багатопроцесорною системою. Мова програмування дозволяє явно вказувати сигнали, які запускають процес. Для запуску процесу обчислень нейроном використовується вхідний сигнал цього нейрона.

Спершу була побудована мережа з двома вхідними величинами, які ідуть на один нейрон (рисунок 6.1) і отримані показники займаного ресурсу ПЛІС та часу максимальної затримки шляху.

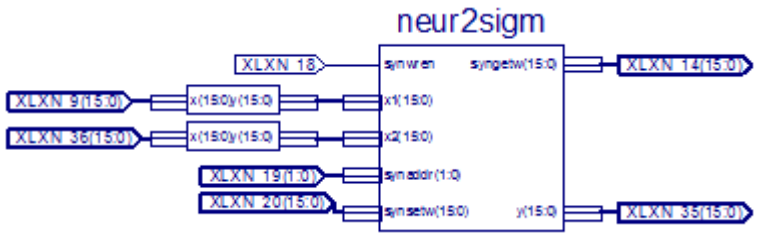


Рисунок 6.1 – Один нейрон з двома входами

Дана мережа з одним нейроном зайняла 695 вентилів логічної матриці LUTs та час максимальної затримки 90.043нс. Далі відповідно було побудовано нейронні мережі з двома входами і двома нейронами (рисунок 6.2), трьома нейронами (рисунок 6.3) і чотирма (рисунок 6.4)

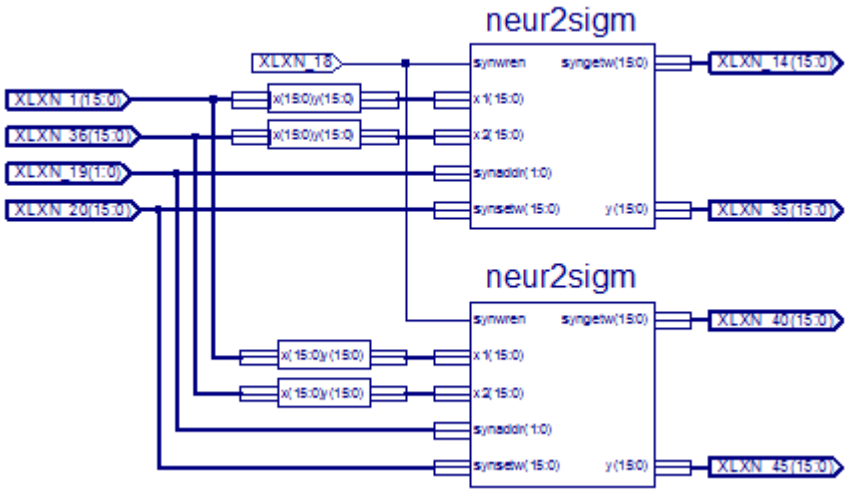


Рисунок 6.2 – Два нейрона з двома входами

Мережа з двома нейронами зайняла 1110 LUTs, що майже вдвічі більше за попередню.

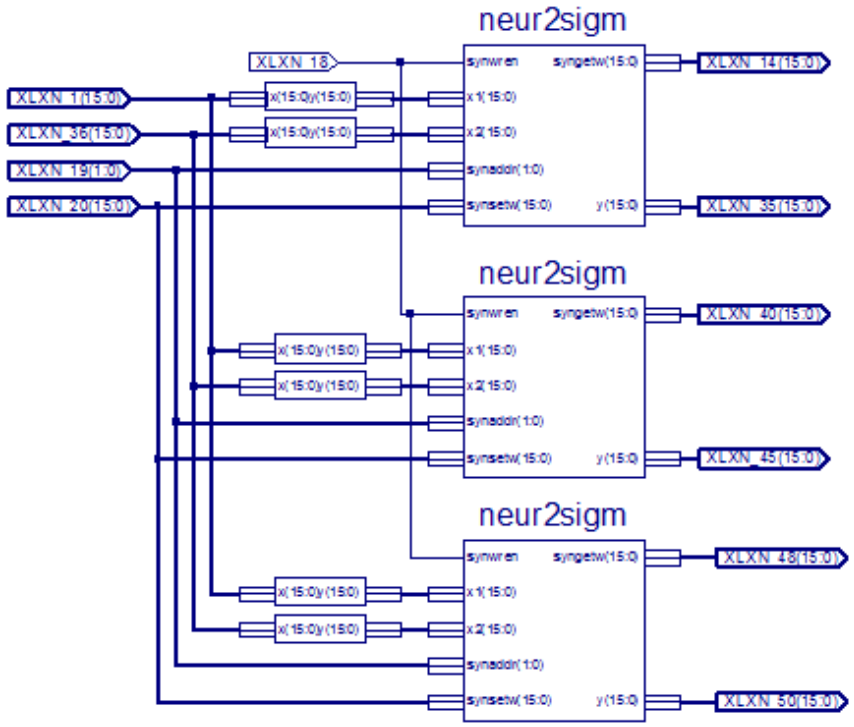


Рисунок 6.3 – Три нейрона з двома входами

Мережі з трьома входами і чотирма займають 1525 LUTs і 1940 відповідно, що показує пряму пропорційність ресурсу ПЛІС від кількості нейронів. Тоді як їх швидкодія 90.133 і 90.141нс.

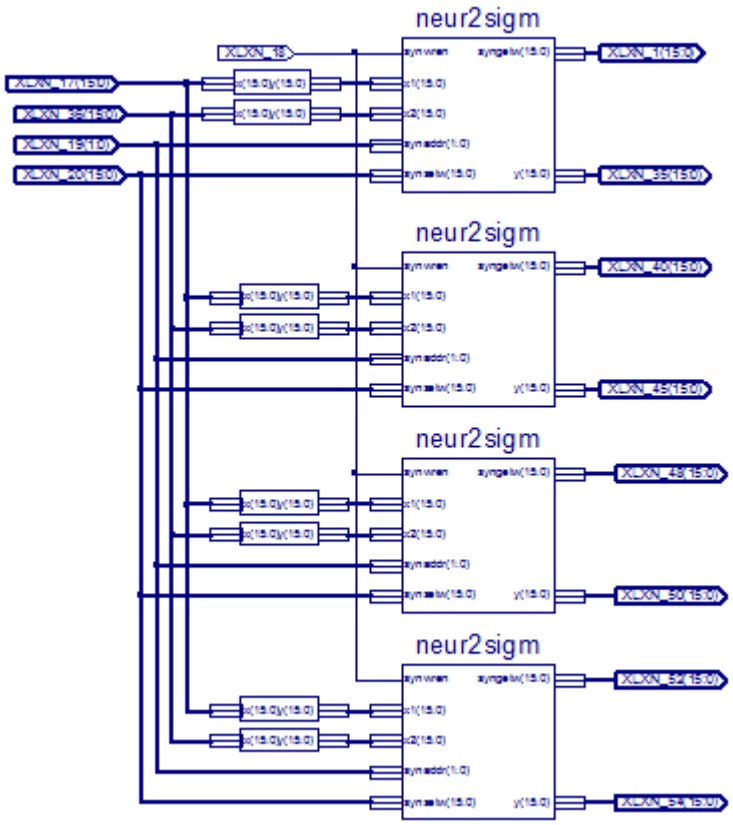


Рисунок 6.4 – Чотири нейрона з двома входами

На рисунку 6.5 зображений графік залежності LUTs та швидкодії мереж в залежності від кількості нейронів з двома входами.

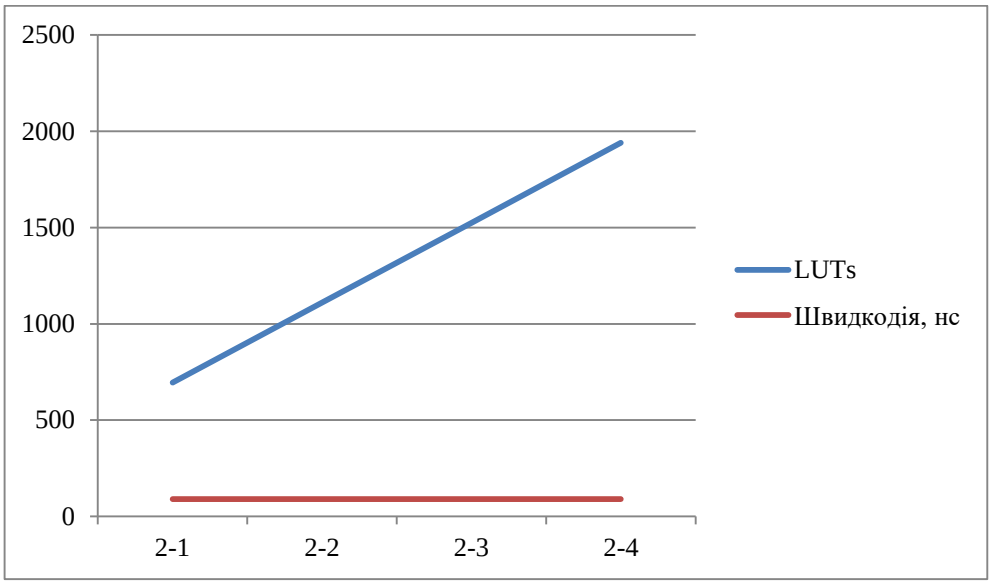


Рисунок 6.5 – графік залежності LUTs та швидкодії мереж з двома вхідними векторами

Як видно з графіка 6.5 ресурс ПЛІС залежить від кількості нейронів, тоді як час лишається незмінним. Такий же дослід був проведений для нейронів з іншою кількістю входів, на рисунку 6.6 можна побачити аналогічний графік для чотирьох входів.

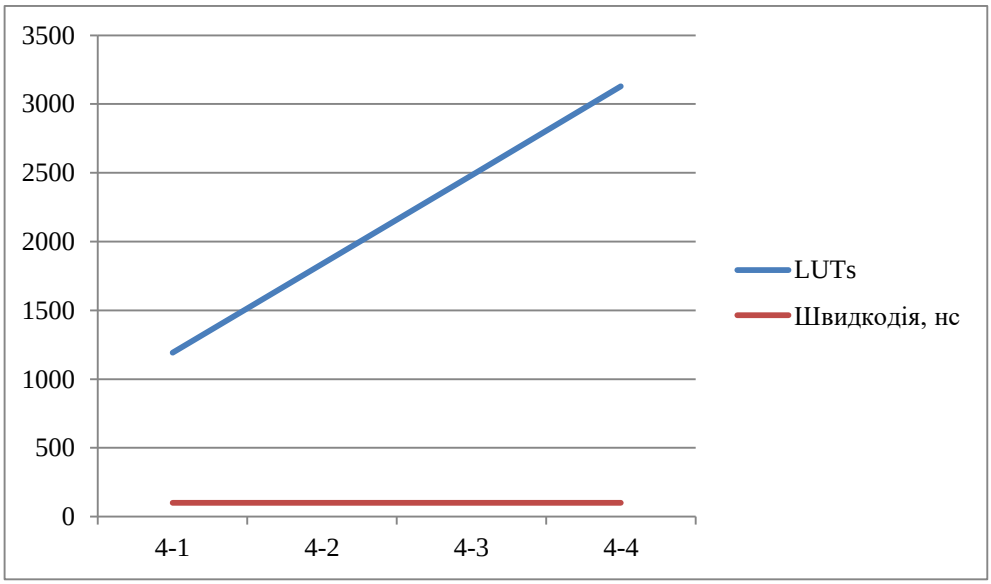


Рисунок 6.6 – графік залежності LUTs та швидкодії мереж з чотирма вхідними векторами

В роботі були апаратно реалізовані повнозв'язні нейронні мережі прямого розповсюдження, і відповідно займаний ними ресурс ПЛІС – LUTs та швидкодія представленні в таблиці 2.1 Топологія нейронних мереж представлена двома числами, де перше – кількість входів на нейрон, друге – кількість нейронів. При конфігуруванні ШНМ прямого розповсюдження використано чіп сімейства Spartan 3 – XC3S5000.

Таблиця 2.1. – Результати моделювання нейронних мереж в ПЛІС

Нейронна мережа	Кількість функцій Гауса	LUTs	Швидкодія, нс
1-1	1	452	86.671
1-2	2	771	86.736
1-3	3	1090	86.761
1-4	4	1409	86.770
2-1	2	695	90.043
2-2	4	1110	90.107
2-3	6	1525	90.133
2-4	8	1940	90.141
3-1	3	942	93.913
3-2	6	1470	93.978
3-3	9	1998	94.003
3-4	12	2526	94.012
4-1	4	1193	101.579
4-2	8	1838	101.755
4-3	12	2483	101.78
4-4	16	3128	101.789
5-1	5	1445	105.123
5-2	10	2192	105.188

Продовження таблиці 2.1

6-1	6	1679	107.54
6-2	12	2511	107.604
8-1	8	2156	112.968
8-2	16	3196	113.033

Деякі реалізовані інші мереже представлені у додатках Е і Є.

Далі на рисунках і можна побачити графіки залежностей ресурсу ПЛІС в залежності від кількості входів на один нейрон і на два.

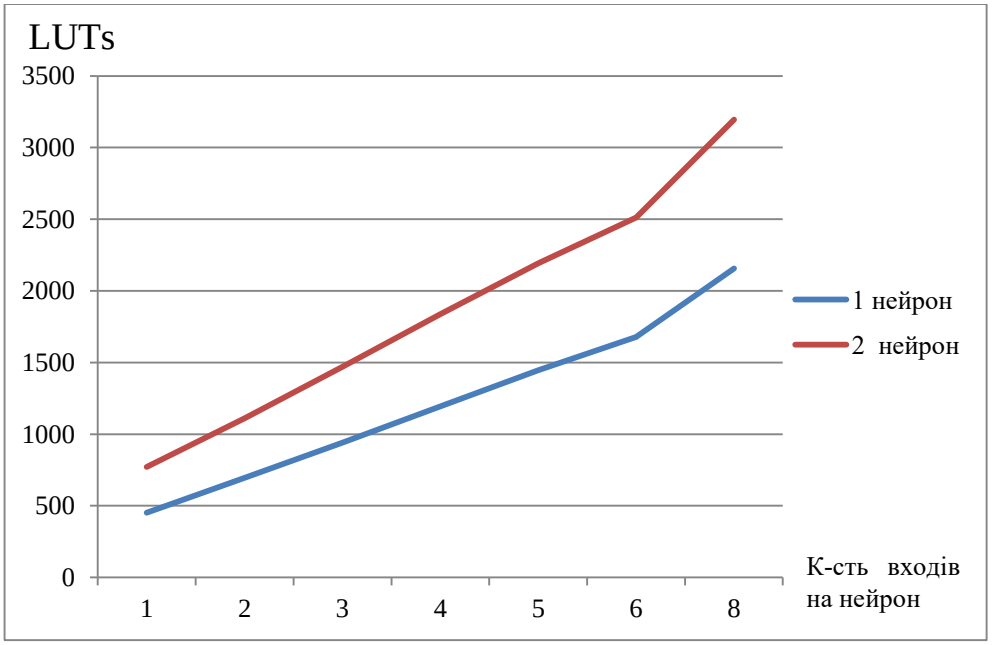


Рисунок 6.7 – Графік залежності LUTs від кількості вхідних векторів

Дані залежності параметрів від кількості нейронів та від кількості вхідних векторів зображені у додатках Ж та З.

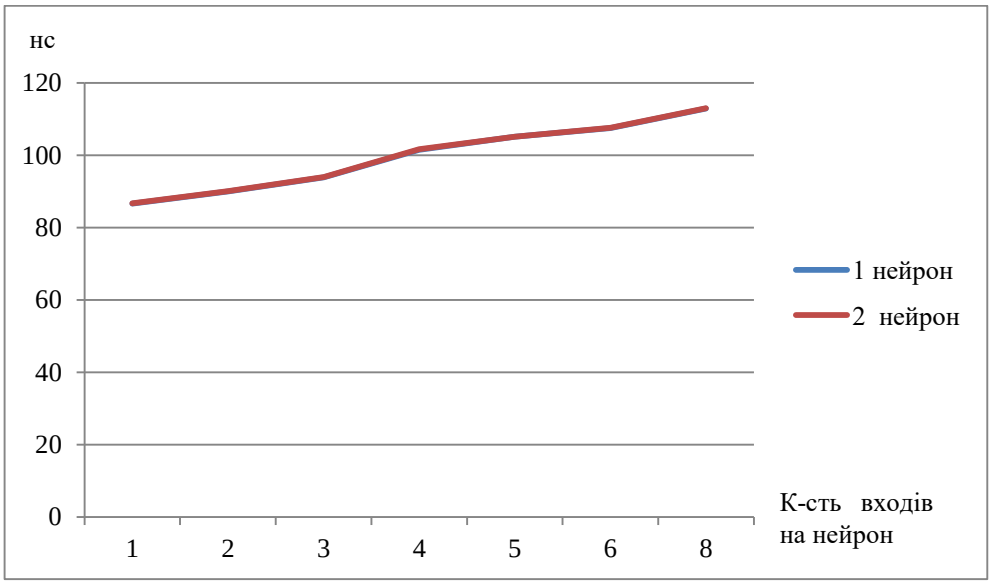


Рисунок 6.8 – Графік залежності швидкодії від кількості вхідних векторів

Отже, зі збільшенням входів на нейрон, а відповідно зі збільшенням кількості функцій Гауса лінійно збільшується кількість LUTs.

7. РОЗРОБКА СТАРТАП-ПРОЕКТУ

7.1 Опис ідеї проекту

За даною роботою був розроблений стартап-проект. Ідеї даного стартап-проекту представлені у таблиці 7.1.

Таблиця 7.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Реалізація радіально-базисних нейронних мереж на ПЛІС	1. Змога подавати паралельно вхідну інформацію	Висока швидкодія
	2. Здатністю до навчання нейронних мереж	Ця технологія корисна для створення пристроїв управління в автоматичних системах

Аналіз сильних, слабких та нейтральних сторін проекту наведено у таблиці 3.2. Там перераховано техніко-економічні властивості проекту, а також розглянуті конкуренти, один з яких було описано у другому розділі.

Таблиця 7.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Реалізація штучних нейронних мереж на багатоядерних процесорах SEAFORTH	Реалізація радіально-базисної нейронної мережі на масивно-паралельній архітектурі графічного процесора			
1.	Швидкодія	80-120нс	700нс	580нс	-	-	+
2.	Ціна	50-100$	50$	50-500$	-	+	-
3.	Похибка	0.5%	2%	0.2-2%	-	-	+

7.2 Технологічний аудит ідеї проекту

У даному підрозділі проведено аудит проекту. Аналіз складових проекту та визначення технологічної здійсненності ідеї зображено в таблиці 7.3.

Таблиця 7.3 – Визначення сильних, слабких та нейтральних характеристик

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
	Реалізація радіально-базисних нейронних мереж на ПЛІС	FPGA (Field-Programmable Gate Array) – Програмована користувачем вентильна матриця, ПКВМ	Наявна	Доступна
		CPLD (A complex programmable logic device) – Комплекс програмованого логічного пристрою	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: FPGA				

7.3 Аналіз ринкових можливостей запуску стартап-проекту

Спершу було проведено аналіз попиту, його наявність, обсяг та динаміка розвитку ринку зображені в таблиці 7.4

Таблиця 7.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	2
2	Загальний обсяг продаж, грн/ум.од	1000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Недискримінаційні якісні
5	Специфічні вимоги до стандартизації та сертифікації	-
6	Середня норма рентабельності в галузі (або по ринку), %	85%

Ринок є дуже привабливим і готовий для сприйняття продукту.

Групи клієнтів та вимоги до товару представлені у таблиці 7.5. Серед аудиторії можуть бути підприємства, які виготовляють або використовують прилади з нейронними мережами, тому переважна кількість це IT компанії. А також споживачами можуть бути ті, хто займаються наукою, тому було записано дослідні інститути.

Таблиця 7.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	Реалізація дешевих та швидких радіально-базисних нейронних мереж	IT компанії, дослідні інститути	Низька цінова політика, доступність до бюджетних компаній; пришвидшення роботи при використанні широких нейронних мереж	Безперервна робота без втручання кваліфікованих працівників

Коли відомі групи споживачів варто проаналізувати фактори, що можуть вплинути на ринок, а також спрогнозувати можливу реакцію компанії, для пристосування до умов. Всі фактори розділені на фактори загроз(таблиця 7.6) та фактори можливостей(таблиця 7.7).

Таблиця 7.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Недовіра споживачів	Так як споживачі це компанії, вони будуть дивитись на нового гравця з опаскою	Спершу пропонувати унікальні пропозиції з безкоштовними тестовими варіантами
2	Задачі, з якими не справляються радіально-базисні нейронні мережі	Унікальні вимоги до задач, які повинен виконувати пристрій з нейронною мережою	Розширення типів мереж для виконання унікальних задач

Таблиця 7.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Ширше коло споживачів	Зацікавленість дрібних промислових підприємств	Відразу налаштовувати виробництво та хапати бика за яйця
2	Ріст промисловості	З’являються нові підприємства	Активний пошук нових можливих майбутніх користувачів

Надалі було проведено аналіз пропозиції, варто визначити загальні риси конкуренції. Цей аналіз представлений у таблиці 7.8.

Таблиця 7.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції – олігополія	Декілька потужних компаній, які мають велику ринкову силу	Можна співпрацювати з конкурентами
2. За рівнем конкурентної боротьби – міжнародний	Розміщення споживачів на земній кулі не має значення	Надавати пропозицію у більшості країн світу
3. За галузевою ознакою – міжгалузева	Боротьба відокремлених товаровиробниками різних галузей економіки	Займати галузі з меншим прибутком, бо там легша конкуренція у поєднанні з ціновою перевагою дасть велику перевагу
4. Конкуренція за видами товарів – товарно-родова	Конкуренція між різними товарами, що виконують схожі функції	Це краща стратегія, бо у товарно-видовій конкуренції складніше
5. За характером конкурентних переваг – цінова	Боротьба за споживачів шляхом пониження ціни	Здешевлення всіх етапів впровадження
6. За інтенсивністю – марочна	Марка не має ролі	Відсутній

Більш детальний аналіз умов конкуренції показаний у таблиці 7.9. Це аналіз стратегії бізнесу за Портером, у якій фігурують п’ять основних сил.

Таблиця 7.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Google, NVIDIA, SEAFORTH	Досягнення певного рівня	Відсутні	Бюджетна реалізація потреб	Відомість та надійність
Висновки:	Індекс концентрації $CR_3 = 50\%$ Ринок помірно концентрований	Прямих можливих конкурентів не було знайдено	Відсутні	Клієнти диктують умови. Ціна, точність роботи	Обмежень немає

Варто розуміти, що розвиваючись в IT сфері основні конкуренти можуть бути одночасно партнерами.

Основні фактори конкурентоспроможності, які були представлені у таблиці 7.2, після проведення подальшого аналізу конкуренції зібрані та обґрунтовані у таблиці 7.10.

Таблиця 7.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Швидкодія роботи	Можливість розпаралелювати роботу нейронних мереж підвищує швидкодію а також може зменшити час навчання нейромереж

Продовження таблиці 7.10

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
2	Цінова політика	Ціна програмованих логічних інтегральних схем значно нижча за інші аналоги, які можуть виконувати дані задачі
3	Точність обчислень	Кусково-лінійна апроксимація забезпечує високу точність при високій швидкодії

За даними факторами була побудована порівняльна таблиця 7.11, де знаком «+» показано відношення рейтингу товарів-конкурентів до запропонованого.

Таблиця 7.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			–3	–2	–1	0	+1	+2	+3
1	Швидкодія роботи	16	-	-	-+	-	-	-	-
2	Цінова політика	18	-	+	-	-	-	-	-
3	Точність обчислень	14	-	-	-	+	-	-	-

У таблиці 7.12 представлено SWOT-аналіз, без якого будь-який аналіз ринкових можливостей не може бути завершеним. Там усі чинники, які з’являються після реалізації, розділені на чотири чотири категорії: сильні сторони(Strengths), слабкі сторони(Weaknesses), можливості(Opportunities), та загрози(Opportunities).

Таблиця 7.12 – SWOT-аналіз стартап-проекту

Сильні сторони: Цінова політика, що гарантує доступність для бюджетних компаній; висока швидкість роботи при малій ціні, точність обчислень	Слабкі сторони: Непривабливість нового гравця на ринку та недовіра через це
Можливості: Розширення кола споживачів за рахунок підприємств, що розвиваються	Загрози: Крадіжка інтелектуальної власності

Проаналізувавши усіх чинників і явищ було розроблені альтернативи ринкового впровадження, які представлені у таблиці 7.13.

Таблиця 7.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Імовірність отримання ресурсів	Строки реалізації
1	Пошук партнерів суміжної галузі	Імовірне	6 місяців
2	Тестове безкоштовне впровадження	Імовірне	3 місяці

З обраних альтернатив було обрано тестове безкоштовне впровадження, так як на імовірність це не впливає, але час реалізації значно коротший. Проте основна альтернатива буде першим кроком і на ній одній впровадження не буде завершене.

7.4 Розроблення ринкової стратегії проекту

Для розроблення стратегії треба першим кроком визначити цільові групи потенційних споживачів. Вони представлені у таблиці 7.14.

Таблиця 7.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	ІТ компанії	Середня	50%	Висока	Дуже складно
2	Дослідницькі інститути	Середня	30%	Низька	Середня складність
3	Технологічні компанії	Середня	70%	Висока	Дуже складно
Які цільові групи обрано: Дослідницькі інститути Технологічні компанії					

За обраними групами цільової аудиторії було вибрано базову стратегію розвитку та описано в таблиці 7.15.

Таблиця 7.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Співпраця з дослідними інститутами на початковому етапі з можливістю подальшого вливання компанії в якусь олігополію	Концентрування на потребах одного цільового сегменту	Низька ціна впровадження та швидша робота пристроїв	Стратегія спеціалізації

Вибір стратегії конкурентної поведінки представлено в таблиці 7.16

Таблиця 7.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Так	Шукати нових	Не буде	Стратегія заняття конкурентної ніші

У таблиці 7.17 представлено стратегії позиціонування, які полягають у тому, що формується ринкова позиція як комплекс асоціацій для споживача.

Таблиця 7.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Явна перевага у швидкодії, незначна ціна	Стратегія спеціалізації	Низька ціна впровадження та швидша робота пристроїв	Нейронні мережі для всіх, дешевизна їх реалізації, висока швидкодія та точність

7.5 Розроблення маркетингової програми стартап-проекту

Коли сформована конкурентоспроможність товару, починається формування маркетингової концепції товару(таблиця 7.18).

Таблиця 7.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Реалізація дешевих та швидких радіально-базисних нейронних мереж	Висока швидкодія, низька ціна	Менший час затримки на одне проходження мережі, зведення похибки до мінімальної

Трирівнева модель товару представлена у таблиці 7.19.

Таблиця 7.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Реалізація дешевих та швидких радіально-базисних нейронних мереж</td></tr>
<tr><td rowspan="3">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Швидкодія
2. Похибка
2. Ресурс ПЛІС</td><td></td><td></td></tr>
<tr><td colspan="3">Якість: похибка до 0.5%</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">До продажу: встановлення архітектури потрібної мережі</td></tr>
<tr><td colspan="3">Після продажу: збір інформації про роботу мереж для можливого подальшого їх покращення</td></tr>
<tr><td colspan="4">Товар буде захищено від копіювання за рахунок патенту на корисну модель</td></tr>
</table>

Після опису моделі товару варто стало можливим визначення меж встановлення ціни, які представлені в таблиці 7.20.

Таблиця 7.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	-	50-500$	1000-1000000$	40-200$

Формування системи збуту є важливим компонентом розробки маркетингової компанії, це представлено у таблиці 7.21.

Таблиця 7.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Через відкриті тендери, вести окремі перемовини з можливими споживачами	Функції інформування та встановлення контакту	Канал одного рівня	Ексклюзивний розподіл

Остання складова маркетингової програми знаходяться в таблиці 7.22. Тут представлена концепція комунікацій.

Таблиця 7.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуютьс я цільові клієнти	Ключові позиції, обрані для позиціонуванн я	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Діяльність на світовому рівні	Прямі офіційні	Послідовна реалізація обраної позиції, унікальність послуги	Заявити про таку пропозицію та доступність для широкого кола споживачів	Раціоналістичн ий тип рекламного звернення

7.6 Висновки

Під час аналізу стартап-проекту було проведено технологічний аудит, проаналізовано ринкові можливості і ринкові стратегії та розроблено маркетингову програму. Було вияснено можливих користувачів, та маркетингова програма підлаштована до відповідної аудиторії.

Рентабельність ринку висока, попит також є, це дає можливість ринкової комерціалізації проекту.

Проект більш науковий і новий, варто співпрацювати з дослідними інститутами, тому перспективи впровадження існують. Хоча в даній сфері дуже сильні конкуренти, це зменшує перспектив, бо так як деякі конкуренти можуть бути майбутніми споживачами, адже вигідно продати проект – це також успішна стратегія.

ВИСНОВКИ

В магістерській дисертації була вирішена задача підвищення швидкодії роботи та покращення точності обчислень радіально-базисних нейронних мереж та їх активаційних функції, розпаралелишви обчислення. Це було досягнуто завдяки розробці нового алгоритму апаратної реалізації на ПЛІС функцій активації Гауса прихованого шару нейронної мережі радіально базисних функцій, використовуючи при цьому мінімальний використаний обчислювальний ресурс.

За даним алгоритмом було реалізовано для прикладу декілька нейронних мереж з різною кількістю вхідних векторів та різною кількістю нейронів. Абсолютна похибка становила 0.005. Нейрони кількістю входів від одного до восьми зайняли від 452 до 2156 LUTs відповідно. У той же час їх швидкодія складала 86.671 та 112.968 нс. Наочно видно, як від кількості функцій активації прихованого шару впливають на роботу мережі. Якщо взяти більшу кількість нейронів, то час проходження суттєво не змінюється за рахунок паралельних обчислень, але кількість LUTs зростає прямо-пропорційно. При реалізації можна зменшити або збільшити кількість лінійних відрізків, можна пожертвувати точністю для ще вищої швидкої та зменшенню ресурсу ПЛІС та навпаки.

Продемонстрована апаратна реалізація радіально-базисної нейронної мережі та її активаційних функцій з даною швидкодією дозволить використовувати їх обчислювальним системам реального часу, які можуть керувати швидкодіючими об’єктами. А саме такі обчислювачі зможуть застосовуватися у безпілотній авіації або для вирішення задач управління автопілотом наземного транспорту. Крім цього вони можуть бути використані і в інших не таких вимогливих системах, де потрібно вирішити задачі управління, апроксимації, розпізнавання та просто класифікувати отримані з відеокамер та датчиків дані.

СПИСОК ДЖЕРЕЛ

1) Терехов В.А., Ефимов Д.В., Тюкин И.Ю. Нейросетевые системы управления, 2002. - 183с.

2) О. К. Колесницкий, к. т. н., доц.; И. В. Бокоцей; А. А. Коренной Анализ принципов построения и свойств устройств для моделирования нейрона, 2012

3) Програмована логіка [Електронний ресурс] – режим доступу до ресурсу: http://um.co.ua/7/7-10/7-107167.html

4) Нейронні структури в виробничих технологіях [Електронний ресурс] – режим доступу до ресурсу: http://lecture.in.ua/lekciya-2-nejronni-strukturi-v-virobnichih-tehnologiyah-civile.html

5) [Електронний ресурс] :http://www.ebay.com/itm/W-Track-Altera-Cyclone-II-FPGA-EP2C5T144-Mini-Development-Learn-Core-Board-E081-/151876594649

6) Multiplayer Perceptron Neural Networks [Електронний ресурс] – режим доступу до ресурсу: https://www.dtreg.com/solution/view/21

7) Миркес Е. М.,Нейрокомпьютер. Проект стандарта. – Новосибирск: Наука, 1999. – 337 с.

8) Радиальные нейронне сети [Електронний ресурс] – режим доступу до ресурсу: http://www.igce.comcor.ru/AI_mag/NN/RadNets/RadNets.html

9) Нейронные сети на основе радиально-симметричных функций [Електронний ресурс] – режим доступу до ресурсу: http://neuronus.com/theory/960-nejronnye-seti-na-osnove-radialno-simmetrichnykh-funktsij.html

10) [Електронний ресурс] : http://www.kit-e.ru/articles/plis/2007_08_175.php

11) Разаев Р.Р., Гоюшов А.И. Интеллектуальная система оценки качества телекоммуникационных услуг – Научный журнал – Информационно управляющие системы, 2014;

12) А. С. Анисимов, А. В. Калачев Реализация искусственных нейронных сетей на многоядерном процессоре SEAFORTH, 2010

13) Савкин Л.В., Дмитриев В.Г., Федоров Е.А. Многослойные персептроны в бортовых системах космической техники: аппаратный подход, 2016,(pdf)

14) Челебаев С.В., Челебаева Ю.А. – Разработка структур преобразователей частотно-временных параметров сигналов в код двух переменных на основе радиально-базисных сетей, 2015

15) Матвеева Н.О. Реализация радиально-базисной нейронной сети на масивно-паралельной архитектуре графического процесора – Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механіки и оптики, 2011,

16) С.А. Лобов, Н.А. Сергиевский, А.А. Харламов Адаптация алгоритма сверточных нейронных се- тей на ПЛИС, 2013

17) Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks, 2012.

18) Порівняння швидкості кількох програмних реалізацій гіперболічного тангенсу [Електронний ресурс] – режим доступу до ресурсу: http://neuroschool.narod.ru/pub/sibzhvm98.html

19) [Електронний ресурс] :http://www.elkos.com.ua/setdsp/xdsp680_face.html

20) Перцептрон [Електронний ресурс] – режим доступу до ресурсу: http://neuropro.ru/memo17.shtml, 2006

21) Bhaskar Bateja, Pankaj Sharma FPGA implementation of artificial neural networks

22) Кравець П.І., Шимкович В.М., Зубенко Г.А., Технологія апаратно-програмної реалізації штучного нейрона та штучних нейронних

мереж засобами FPGA – Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка, 2012

23) Сергієнко А.М., Сергієнко П.А. Реалізація функції квадратного кореня у ПЛІС, 2008

24) Alexander Gomperts, Abhisek Ukil, Franz Zurfluh Implementation of Neural Network on Parameterized FPGA, 2010

25) Georgios Smaragdos, Sebastian Isaza, Martijn Van Eijk, Ioannis Sourdis, Christos Strydis FPGA-based Biophysically-Meaningful Modeling of Olivocerebellar Neurons 2014

26) Нейронні структури в виробничих технологіях цивільної авіації [Електронний ресурс] – режим доступу до ресурсу: http://lecture.in.ua/lekciya-2-nejronni-strukturi-v-virobnichih-tehnologiyah-civile.html

27) Кравец П.И., Шимкович В.Н., Ференс Д.А. Электронное моделирование, 2015,

Додаток А

Доповідь з виступу на конференції

ALGORYTM IMPLEMENTACJI PROMIENIOWO-PODSTAWOWYCH SIECI NEURONOWYCH I ICH FUNKCJI AKTYWACYJNYCH NA FPGA

ALGORITHM FOR THE IMPLEMENTATION OF RADIAL-BASIC NEURAL NETWORKS AND THEIR ACTIVATION FUNCTIONS ON THE FPGA

Streszczenie

W artykule opracowano i opisano algorytm implementacji komponentu sprzętowego na bazie FPGA w języku VHDL, który implementuje sztuczny nerw promieniowy-podstawowej sieci neuronowej z funkcjami aktywacji Gaussa w ukrytej warstwie i zbadano nowe elementy sprzętu.

Słowa kluczowe: Sieć RBF, FPGA, VHDL, funkcje aktywacji

Abstract

In the article the algorithm of implementation of the hardware component on the basis of FPGA in the language of VHDL, which implements the artificial neuron of the radial-basic neural network with Gaussian activation functions in the hidden layer, was developed and described, and new hardware components were investigated.

Keywords: RBF-network, FPGA, VHDL, activation functions

[1] Department of Automation and Control in Technical Systems
National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute»
Kyiv, Ukraine

1. Introduction

Neural network control systems represent a new high-tech direction in control theory and relate to a class of nonlinear dynamical systems [1-3]. The ability to provide parallel input information gives high performance, and combined with the ability to teach neural networks makes this technology useful for creating control devices in automatic systems [4-8].

RBF networks are universal approximators and can be used for approximation of any continuous function with minor constraints [9, 10]. RBF networks in their structure refer to two-layer networks, which use a hidden layer with a fixed non-linear transformation of the input vector with constant weight coefficients. This layer carries a static display of input variables $\mathbf{r} \in R^n$ in new variables $\mathbf{q}_1^{(l)} = col(q_1^{(l)}, ..., q_m^{(l)})$. The second, linear output, layer "weighs" these variables with adjustable weights, $\mathbf{w}_i = col\left(w_{i,1}, w_{i,2}, ..., w_{i,m}\right)$, so that, for example, the *i*-th scalar output of the RBF network is written as:

$$q_i^{(2)} = q_i = F(\mathbf{r}) = \sum_{j=1}^{m} w_{i,j} f_i(\mathbf{r}) + w_0 \cdot 1, \quad i = 1, 2, ..., K \tag{1}$$

Structure of the RBF network with the input vector $\mathbf{r} \in R^n$ and exit vector $\mathbf{q}^{(2)} = \mathbf{q} \in R^n$ shown in Fig. 1. Activation features $f_j(r): R^n \to R^1$ are selected in the class of «radial» symmetric functions:

$$f_i(\mathbf{r}) = f(\|\mathbf{r} - \mathbf{a}_j^*\|; h), \quad \mathbf{a}_j^* \in R^n. \tag{2}$$

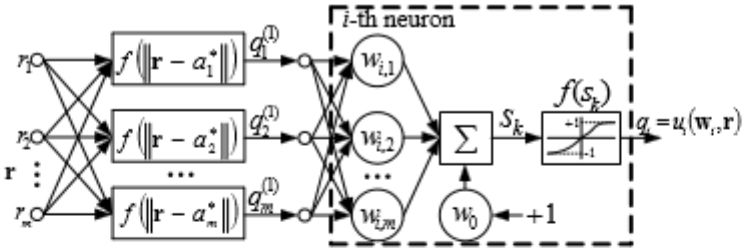


Fig. 1. Block diagram of the RBF network

They have an extremum when changing the inputs **r** only near the established values of the weight coefficients of the hidden layer neurons $\mathbf{w}_j^{(1)} = \mathbf{a}_j^*, j = 1, 2, ..., m;$, $\mathbf{w}_j^{(1)} = col(w_{j,1}^{(1)}, w_{j,2}^{(1)}, ..., w_{j,n}^{(1)})$.

Weights of the second layer $w_{i,j} = w_{i,j}^{(2)}$, the parameter *h*, the position of the so-called «centers of the field of susceptibility», which depend on the established values of weight coefficients in the first layer $\mathbf{w}_j^{(1)} = \mathbf{a}_j^*$, form of functions $f_j(\cdot)$ and their number *a* collectively determine the properties of the RBF network.

As functions of $f(z)$ one of the functions is selected:

$$\exp(-z^2); \quad \exp(-z); \quad 1/(1+z)^2; \; 1/1+z^2; \quad z^n \log z; \quad z^2 + \alpha z + \beta.$$

In practice, the first of these functions, the Gaussian function, is most used:

$$f_j(\mathbf{r}) = \exp\left(\sum_{k=1}^{n} 0.5\sigma_{j,k}^{-2} \left\|\mathbf{r} - \mathbf{a}_j^*\right\|^2\right) \quad (3)$$

where $\mathbf{a}_j^* = col(a^*_{j,1}, a^*_{j,2}, \cdots, a^*_{j,n})$ – a numerical vector, and a $\sigma_{j,k}$ – width of Gaussian function. The specificity of the RBF network is that in the operating mode only the weight coefficients of the linear output layer are adjusted in them. The approximation error is calculated directly on the network output as well as in multilayer neural networks, but the debugging of only one, linear by parameters of the layer debug removes the problem of finding the global minimum of training functions (usually quadratic) and promotes a rapid convergence of the learning process of the network.

The problem with RBF networks is the choice of the number of radial functions necessary for approximation, and this fact becomes a critical factor in the use of RBF networks in identification and control tasks where information required to determine the dimension of RBF networks is generally absent. It is noted that the number of necessary radial-basic functions increases exponentially with the increase in the number of input variables, that is, in our case with the growth of m. Hence the conclusion about the applicability of RBF networks in those problems where the dimension of the input set r is limited and known (there is no such limitation in multilayer networks).

2. Formulation of the problem

As noted above, the RBF-networks are universal approximators and, with slight restrictions, can be applied to approximate any continuous function. RBF networks in their structure refer to two-layer networks, which use a hidden layer with a fixed non-linear transformation of the input vector with constant weight coefficients. This layer carries a static display of input variables $\mathbf{r} \in R^n$ in new variables $\mathbf{q}_1^{(l)} = col(q_1^{(l)}, ..., q_m^{(l)})$. One of the problems with hardware implementation of neural network control systems based on RBF networks is the implementation of an artificial neuron and its nonlinear activation functions by means of FPGA. Existing approaches to the digital realization of nonlinear functions use different methods of approximation, such as tabular method, Taylor series expansion, piecewise linear approximation, etc. Taylor series expansion requires many multiplications, and therefore unacceptable for implementation in FPGA, since the multiplication block occupies a large amount of resources. The tabular method involves the creation of a global variable, a table of possible values of the target function, unpredictable and uncontrolled access to it by all neurons in the network, which in turn creates a large temporary delay, and the creation of a separate local table for each neuron in the network is unacceptable in terms of the use of the FPGA resource. Therefore, the most optimal method for the realization of nonlinear activation functions is a piecewise linear approximation.

The purpose of this article is to develop an algorithm for hardware implementation of FPGA radial-basic neural networks and their activation functions, which will significantly increase the speed of their work, due to parallel computing, with minimal use of computing resource.

3. The problem solution and experimental results

In this work [11, 12] a general method of realization of nonlinear activation functions of artificial neurons has been developed and described, an algorithm for the implementation of an artificial neuron with sigmoid activation function has been developed, hardware neuron on and neural networks have been implemented in FPGA and their research has been carried out.

At the beginning of the research performed the function activation relative symmetry axes. If the condition is executed

$$1 - f(x) = f(-x), \tag{4}$$

then $f(x)$ can be seen only positive argument for a negative - determined by the formula (3), which in turn will accelerate the computation of the function and reduce FPGA resource occupied.

Further, based on the specified accuracy approximation form, a piecewise linear activation function with the appropriate number of intervals and each interval $(-\infty; x_1), (x_1; x_2); \ldots (x_n; +\infty)$ represents a different formula. General description of the piecewise linear function activation record as:

$$f(x) = \begin{cases} k_0 x + b_0, x < x_1 \\ k_1 x + b_1, x_1 < x < x_2 \\ \\ k_n x + b_n, x_n < x \end{cases}, \tag{5}$$

Next to each linear function forward coefficients k and b and remember them. Further calculations for the function $f(x)$ for argument's memory are selected from the corresponding coefficients k and b and the corresponding formula calculated value $f(x)$, and for negative values of the argument function values calculated by the formula 3.

In the hardware implementation of the RBF network, one of the main problems is the implementation of the hidden layer and the nonlinear activation functions $f_i(r)$, in particular the implementation of the Gaussian function.

The Gaussian function is shown in Figure 2 and is described by the formula:

$$y = ae^{\frac{(x-b)^2}{2\sigma^2}} \tag{6}$$

where a is responsible for the height of the function, b – for moving in the axis x, and σ for the rate of decay function when the vector is removed from the center.

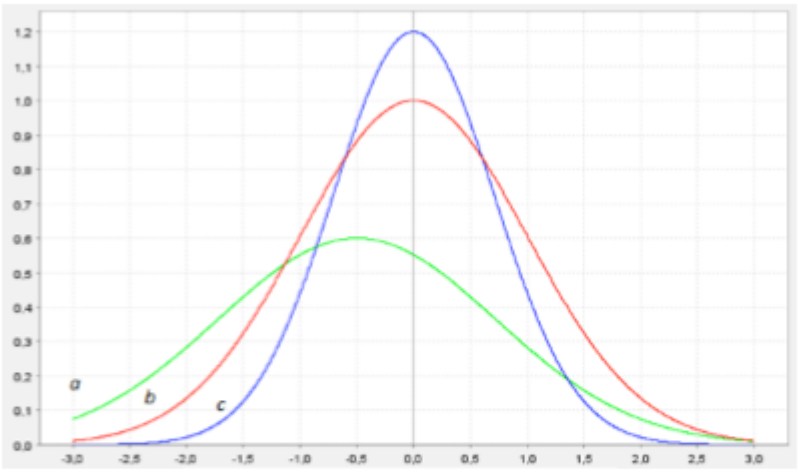


Fig. 2. Graphs of Gaussian functions

The graph has low falling speed and offset along the x-axis and is shown in Figure 2a:

$$y = 0.6e^{-\frac{(x+0.5)^2}{3}} \tag{7}$$

The graph is elongated along the y-axis. It is shown in Figure 2c:

$$y = 1.2e^{-\frac{x^2}{1}} \tag{8}$$

All these functions are selected when training the neuron, depending on the desired result.

We assign the following initial form of the Gaussian function (Figure 2b):

$$y = e^{-\frac{x^2}{2}} \tag{9}$$

We propose the following step-by-step algorithm for hardware implementation of FPGA artificial neuron RBF-network with hidden layer with Gaussian functions. The algorithm is executed as follows.

Step 1. We present the value of the input vector to the inputs of the artificial neuron and set the variable x. Numbers are 16-bit, 9 bits are coming to the integral part and 7 bits are coming to fraction part.

Step 2. Also, we calculate the absolute value from the Gaussian function argument, set the change x':

$$x' = |x| \tag{10}$$

Step 3. At this step, the Gaussian activation function is split into linear pieces, and the coefficients of the linear equations k and b are determined. A breakdown into linear pieces of their equation is shown in Figure 3.

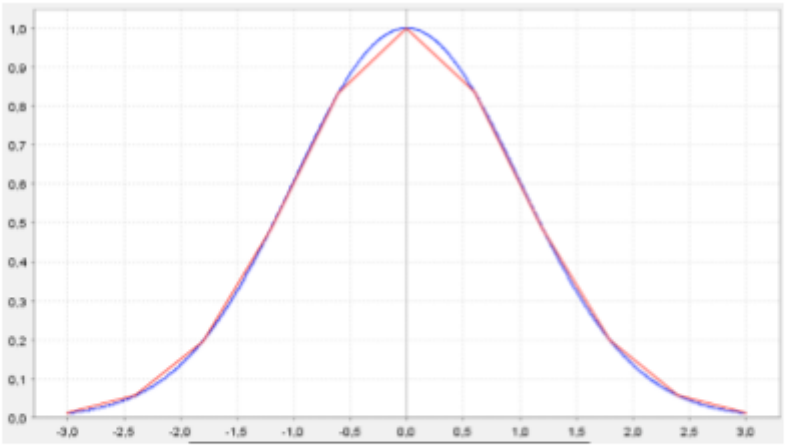


Fig. 3. The piecewise linear approximation of the Gaussian function

Determine the coefficients of the linear levels:

$$k,b=\begin{cases}[-0.275,\ 1], if\ 0\le x'<0.6\\ [-0.58,\ 1.183], if\ 0.6\le x'<1.2\\ [-0.48,\ 1.063], if\ 1.2\le x'<1.8\\ [-0.238,\ 0.628], if\ 1.8\le x'<2.4\\ [-0.093,\ 0.28], якщо\ x'\ge 2.4\end{cases} \tag{11}$$

Step 4. Determine the variable f, calculate the value of the formula:

$$f=k\cdot x'+b \tag{12}$$

Step 5. The result f is fed to neuron with a sigmoid activation function.

The general view of a RBF network with four neurons of a hidden layer and one neuron with a sigmoid activation function is shown in Figure 4.

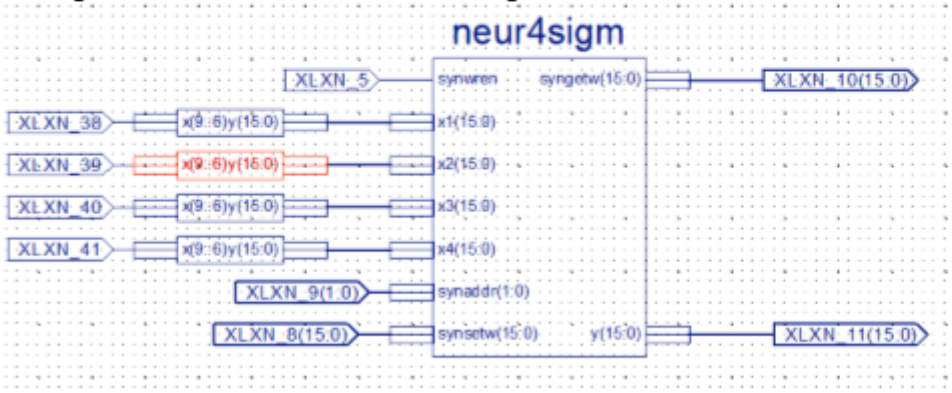


Fig. 4. RBF network hardware is implemented on the FPGA

The computational process implemented by this algorithm, the neural network with 4 neurons of the hidden layer and one neuron with a sigmoid activation function on the programmable logic device using 16-bit numbers with a fixed point took 1089 Look Up Tables. Each hidden layer neuron of the RBF network is developed at FPGA as a separate computing unit, it occupies 93 LUTs. The error in the implementation of the sigmoidal function is absolute ±0.005. The speed, as the total delay of the combinational circuit of the block of the neural network, was 114.1 ns. From these results, it can be seen that the number of LUTs increased by 63% when compared with one neuron of the sigmoid activation function, while the time increased by only 50%, due to the fact that the hidden layer neurons can be calculated in parallel.

4. Conclusion

Hardware realization of radial-basis neural networks with such speed allows them to be used in real-time computing systems in the management of high-speed objects. In particular, such calculators can be used to solve automobile pilot problems, in unmanned aerial vehicles, where they can solve problems of recognition, control, approximation and classification of data received from video cameras and others sensors.

References

[1] S. Omatu, M. Khalid, R. Yusof, *Neuro-Control and Its Applications, Germany*, Berlin:Springer-Verlag, 1997.

[2] G. Dreyfus, *Neural Networks: Methodology and Applications*, SpringerVerlag Berlin Heidelberg. 2005. P. 498.

[3] A.L. Edelen, S.G. Biedron, B.E. Chase, D. Edstrom, S.V. Milton, P. Stabile, *Neural Networks for Modeling and Control of Particle Accelerators*, IEEE Trans.Nucl.Sci. 63 (2016) no.2, pp. 878-897

[4] Sundararajan N. Saratchandran P. Yan Li *Fully Tuned Radial Basis Function Neural Networks for Flight Control*, Springer US. 2002. P. 158.

[5] M. Chen, S.S. Ge, B. Voon Ee How, *Robust Adaptive Neural Network Control for a Class of Uncertain MIMO Nonlinear Systems With Input Nonlinearities*, IEEE Trans.Neural Networks, 2010, 21, pp. 796-812

[6] M. Lawryńczuk, *Computationally Efficient Model Predictive Control Algorithms: A Neural Network Approach,* Springer. Studies in Systems, Decision and Control, 2013. vol. 3.

[7] Oliver Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*, Springer Science & Business Media, 2013. P. 786.

[8] Sigeru Omatu, *Classification of mixed odors using a layered neural network*, International Journal of Computing. 2017, Volume 16, Issue 1. pp. 41-48.

[9] Yuan Xiaofang, Wang Yaonan, Sun Wei, Wu Lianghong, *RBF Networks-Based Adaptive Inverse Model Control System for Electronic Throttle*, IEEE Transactions on Control Systems Technology, 2010, Volume: 18, Issue: 3 pp. 750 – 756.

[10] Tianshu Li, Shukai Duan, Jun Liu, Lidan Wang, *Tingwen Huang, Power Calculation Using RBF Neural Networks to Improve Power Sharing of Hierarchical Control Scheme in Multi-DER Microgrids*, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2016, Volume: 46, Issue: 4, pp. 582 – 588.

[11] Kravets, P.I., Shymkovych. V.M., Zubenko, G.A., *Technology of Hardware and Software Implementation of Artificial Neurons and Artificial Neural Networks by Means of FPGA* in Ukrainian, Visnyk NTUU "KPI" Informatics, operation and computer sciense, 2012, №55, pp. 174-180.

[12] Petro Kravets, Volodymyr Shymkovych, *Neural Network Control System with Direct and Inverse Model of Control Object Hardware and Software Realization of in FPGA,* Information Technology, Computational and Experimental Physics. Kulczycki P., Kowalski P.A., Lukasik S. (eds.) AGN-UST. Krakow, Poland - 2016. - pp. 180-183.

ДОДАТОК Б

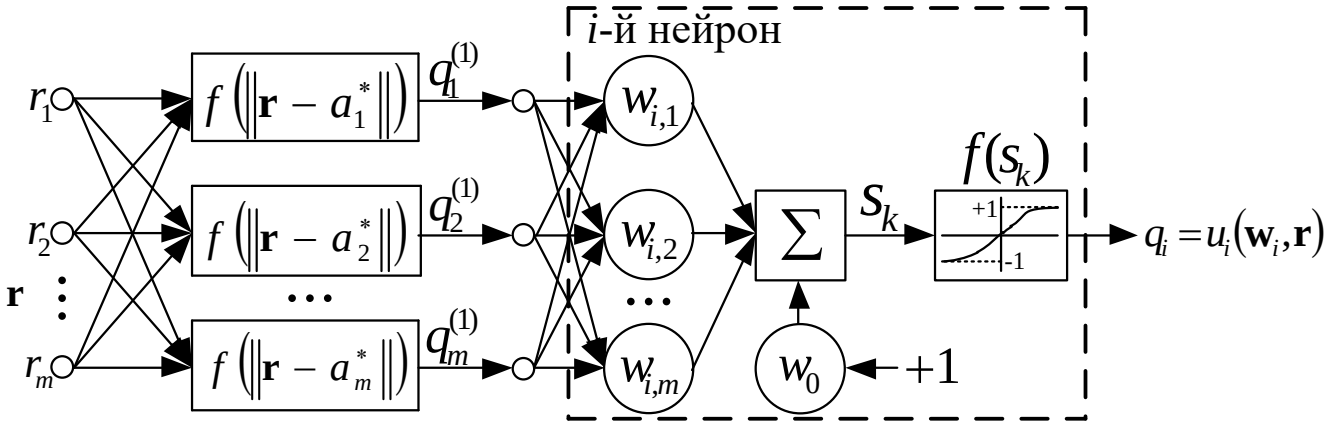
Код програми

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library ieee_proposed;
use ieee_proposed.fixed_pkg.ALL;

library work;
use work.ALL;

entity Gaussian_function is
    Port (x: in STD_LOGIC_VECTOR(15 downto 0);
               y: out STD_LOGIC_VECTOR(15 downto 0));
end Gaussian_function;

architecture Behavioral of Gaussian_function is

begin

process (x)
    variable k, b: sfixed(0 downto -10);
    variable kx: sfixed(11 downto -16);
    variable r_kx: sfixed(11 downto -6);
    variable result: sfixed(12 downto -10);
    variable absx: sfixed(10 downto -6);
    begin

          absx := abs(to_sfixed(x, 9, -6));

          if (absx < to_sfixed(0.6, absx)) then
               k := to_sfixed(-0.275, k);
               b := to_sfixed(1, b);
```

```
        elsif (absx < to_sfixed(1.2, absx)) then
             k := to_sfixed(-0.58, k);
             b := to_sfixed(1.183, b);
        elsif (absx < to_sfixed(1.8, absx)) then
             k := to_sfixed(-0.48, k);
             b := to_sfixed(1.063, b);
        elsif (absx < to_sfixed(2.4, absx)) then
             k := to_sfixed(-0.238, k);
             b := to_sfixed(0.628, b);
        elsif (absx < to_sfixed(3, absx)) then
             k := to_sfixed(-0.093, k);
             b := to_sfixed(0.28, b);
        else
             k := to_sfixed(0, k);
             b := to_sfixed(0, b);
        end if;

        kx := (k*absx);
        r_kx := kx(11 downto -6);

        result := (r_kx + b);

        y <= to_slv(resize(result, 15, 0));

    end process;
end Behavioral;
```

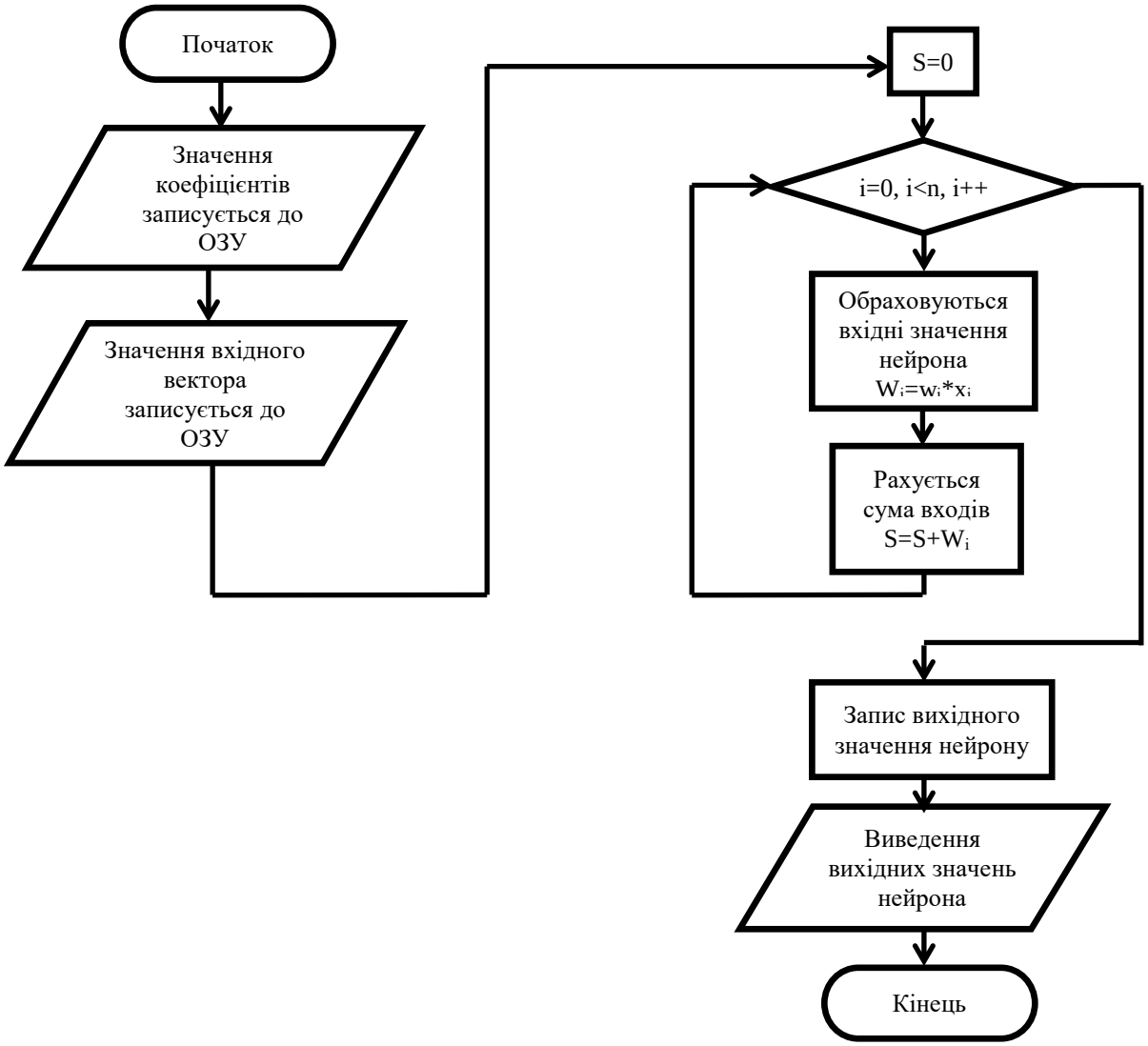
ДОДАТОК В

Структурна схема радіально-базисної нейронної мережі



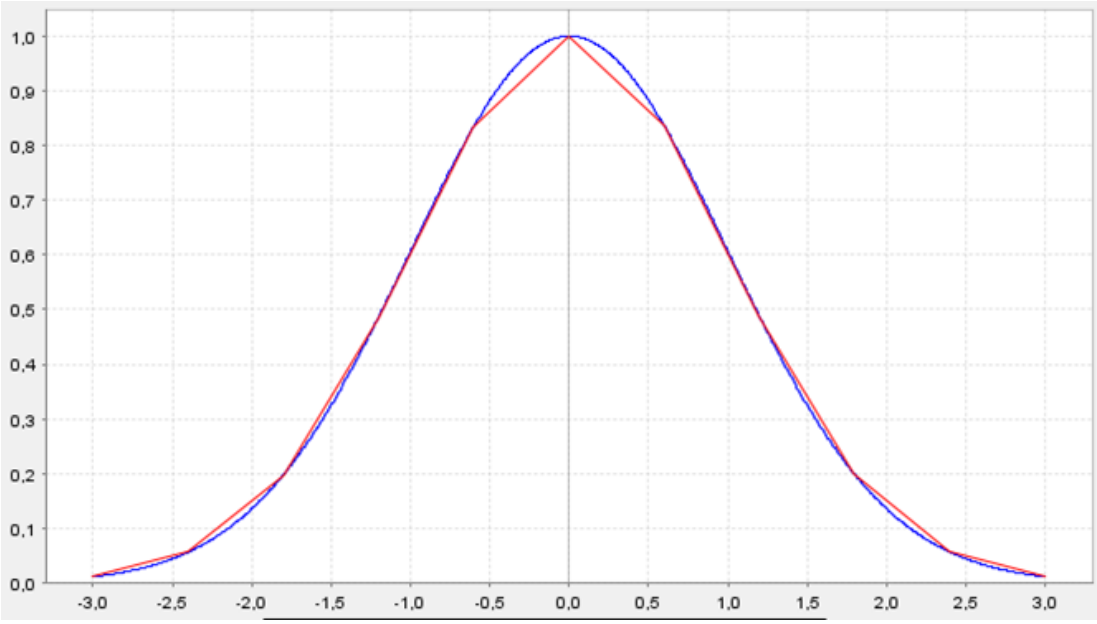
ДОДАТОК Г

Блок-схема алгоритм обчислення вихідного значення нейрона



ДОДАТОК Ґ

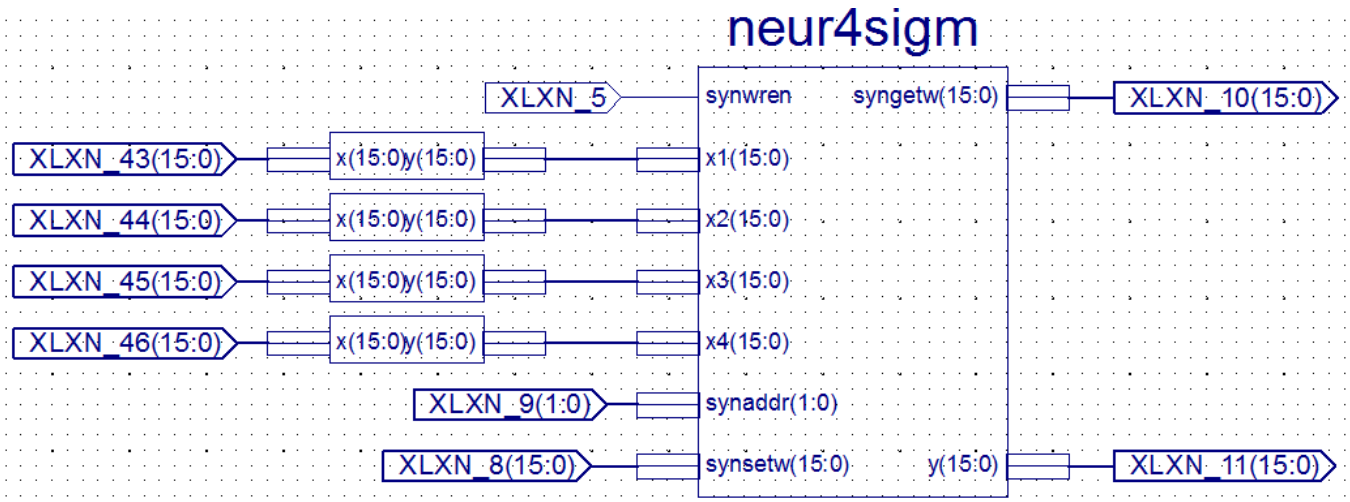
Функція Гауса



$$k, b = \begin{cases} [-0.275, 1], \text{якщо } 0 \leq x' < 0.6 \\ [-0.58, 1.183], \text{якщо } 0.6 \leq x' < 1.2 \\ [-0.48, 1.063], \text{якщо } 1.2 \leq x' < 1.8 \\ [-0.238, 0.628], \text{якщо } 1.8 \leq x' < 2.4 \\ [-0.093, 0.28], \text{якщо } x' \geq 2.4 \end{cases}$$

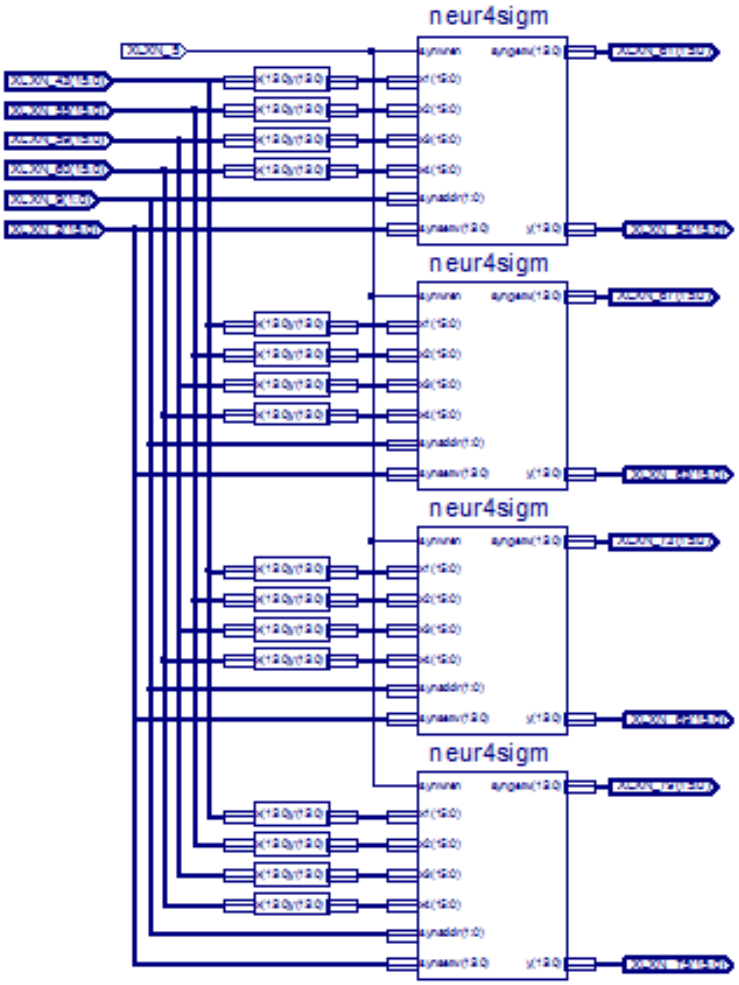
ДОДАТОК Д

Апаратна реалізація радіально-базисної нейронної мережі на ПЛІС



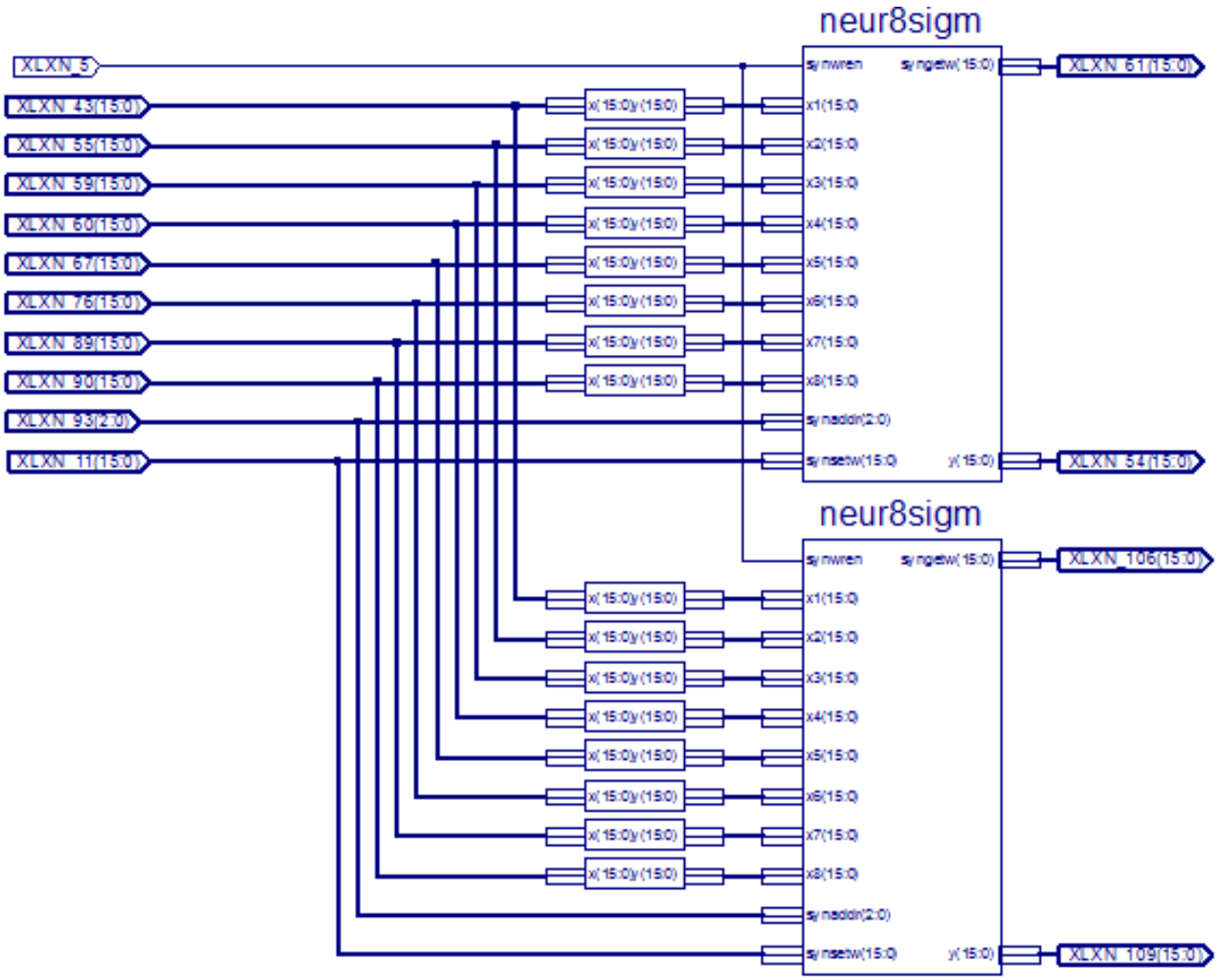
ДОДАТОК Е

Мережа з чотирма вхідними векторами та чотирма нейронами



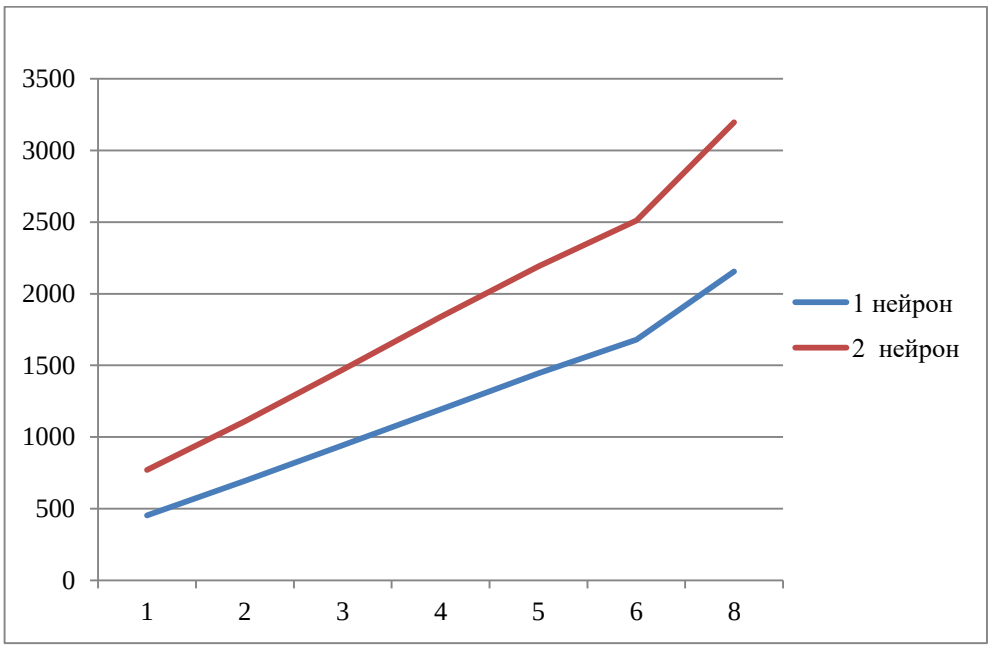
ДОДАТОК Є

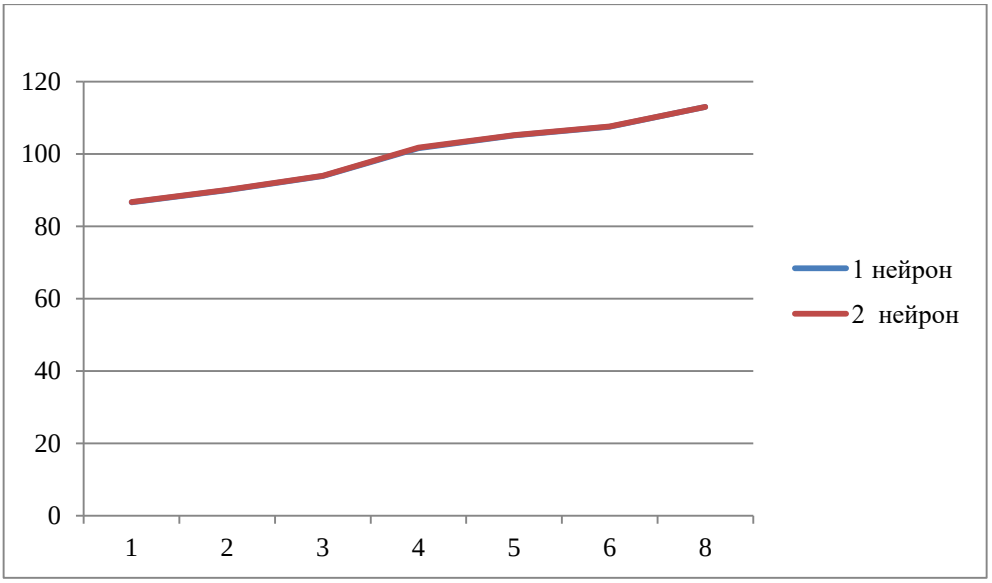
Мережа з восьма вхідними векторами та двома нейронами



ДОДАТОК Ж

Графіки залежностей параметрів від кількості вхідних векторів





ДОДАТОК З

Графіки залежностей параметрів від кількості нейронів